HELSINKI UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMMUNICATIONS ENGINEERING

EERO KELTIKANGAS

ENTERPRISE ARCHITECTURE DOCUMENTATION AND REPRESENTATION

A PRAGMATIC DOCUMENTATION FRAMEWORK

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN ENGINEERING

ESPOO, SEPTEMBER 6, 2006

SUPERVISOR: TOMI MÄNNISTÖ

INSTRUCTOR: PEKKA KÄHKIPURO

AUTHOR:	EERO KELTIKANGAS
NAME OF THE THESIS:	ENTERPRISE ARCHITECTURE DOCUMENTATION AND REPRESENTATION – A PRAGMATIC DOCUMENTATION FRAMEWORK
DATE: SEPTEMBER 6, 2006	*NUMBER OF PAGES*: 45
FACULTY:	DEPARTMENT OF ELECTRICAL AND COMMUNICATIONS ENGINEERING
PROFESSORSHIP:	TIK-76 INFORMATION PROCESSING SCIENCE
SUPERVISOR:	TOMI MÄNNISTÖ, PROFESSOR (PRO TEM), DR.SC. (TECH.)
INSTRUCTOR:	PEKKA KÄHKIPURO, DR.SC. (TECH.)

As a consequence of information technology's pervasive role in businesses nowadays and the rate of change businesses experience in their operating environment, many organizations have outgrown from their former IT management practices. Enterprise architecture is promoted to them as the next solution. Unfortunately, the established enterprise architecture frameworks and especially their documentation conventions are clearly not up to the task, yet. Their approach is rather technically oriented – at least method-wise, if not content-wise too. Consequently, many enterprise architecture initiatives although having produced vast volumes of models and architecture documents, have actually added very little to organization's capability to strategically advance the properties and qualities of its information systems.

This thesis studies the optimal structure and form of enterprise architecture documentation from the point of view of the above capability. As a starting point for the study the hurdles and difficulties faced by two case projects are reported. The presented novel documentation framework, which is the main contribution of this work, is later shown to avoid these complications. It is structured as a mesh of relatively small pieces of documentation, each piece having a distinct physical counterpart. As a documentation form, the framework uses almost exclusively architectural principles.

KEYWORDS:	Enterprise architecture, Architecture documentation, Architectural principles

TEKIJÄ:		EERO KELTIKANGAS
TYÖN NIMI:		YRITYSARKKITEHTUURIN DOKUMENTOINTI JA KUVAAMINEN — KÄYTÄNNÖNLÄHEINEN KUVAUSKEHYS
PÄIVÄMÄÄRÄ:	6.9.2006	*SIVUMÄÄRÄ*: 45
OSASTO:		SÄHKÖTEKNIIKAN OSASTO
PROFESSUURI:		TIETOJENKÄSITTELYOPPI KOODI: TIK-76
TYÖN VALVOJA:		PROFESSORI TOMI MÄNNISTÖ
TYÖN OHJAAJA:		FT PEKKA KÄHKIPURO

Osa yrityksistä on kaikkialle niiden toimintaan leviävän informaatioteknologian ja niiden toiminta-ympäristön muutosvauhdin seurauksena kasvanut ulos aikaisemmista informaatioteknologian johtamiskäytännöistään. Seuraavana ratkaisuna näille yrityksille suositellaan yritysarkkitehtuuria. Valitettavasti vakiintuneet yritysarkkitehtuurikehykset ja etenkin niiden dokumentointikäytännöt eivät vielä ole riittävällä tasolla. Niiden lähestymistapa on melko teknisesti orientoitunut – ainakin menetelmämielessä ellei myös sisällön suhteen. Tästä johtuen monet yritysarkkitehtuurihankkeet, vaikka olisivat tuottaneet valtavan määrän malleja ja arkkitehtuuridokumentaatiota, ovat käytännössä lisänneet hyvin vähän organisaation kykyä strategisesti kehittää sen informaatiojärjestelmien ominaisuuksia ja laatua.

Tämä työ tutkii yritysarkkitehtuuridokumentaation optimaalista rakennetta ja muotoa edellä mainitun kyvyn kannalta. Tutkimuksen lähtökohdaksi esitellään kahden case-projektin kohtaamia esteitä ja vaikeuksia. Esiteltävän uudenlaisen dokumentointikehyksen, joka on tämän työn pääasiallinen kontribuutio, osoitetaan myöhemmin välttävän kyseiset hankaluudet. Sen rakenne on suhteellisen pienten dokumentaation osien verkko, jossa kullakin osalla on selkeä fyysinen vastine. Dokumentointimuotona kehys käyttää lähes yksinomaan arkkitehtonisia periaatteita.

AVAINSANAT:	Yritysarkkitehtuuri, arkkitehtuurin dokumentointi, arkkitehtoniset periaatteet

ACKNOWLEDGEMENTS

This has not been an easy journey, although I doubt it ever is. During the crafting period of this thesis I have been actively involved in the design and development of two major, multi-year IT projects and our third child, Antti, was born. As anyone can expect, spare-time and mental energy had already otherwise been low, yet this effort required a considerable amount of both. However, now that the journey is finally over, I would like to acknowledge the following individuals.

Pekka Kähkipuro and Jussi Ollikainen for commenting, challenging and plainly helping me to form the framework. Pekka Kähkipuro again together with Tomi Männistö for their constructive feedback on this thesis and for tolerating my usually challenging timetables. All unnamed colleagues with whom I have worked in enterprise architecture related assignments for exchanging ideas and helping me to polish the framework.

Last but certainly not least, my dear wife for constantly encouraging and pushing me through by helping in every conceivable way.

Espoo, September 6, 2006

Eero Keltikangas

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

CHAPTER I
INTRODUCTION

> ***architecture*** *The art and science of designing buildings and other structures.*
>
> *(Wikipedia)*

1.1 OVERVIEW

It is widely recognized that the role of corporate IT (Information Technology) has transformed radically over the last decade (see, for example, Hirvonen, 2005). Today it serves not only to support business strategy, but increasingly and more importantly to drive and shape it – and vice versa (Ross, 2003, and Malan and Bredemeyer, 2005). This strategically much more salient role has inspired plenty of both academic and industry effort around the notion of *enterprise architecture*.

Unfortunately, there is no shared definition of enterprise architecture. Malan and Bredemeyer (2005) explain that with the evolution of the notion – "enterprise architecture has evolved during the past decade from enterprise architecture as technology architecture (EA = TA), to enterprise architecture as enterprise-wide IT architecture (EA = EWITA), to enterprise architecture as the architecture of the enterprise, encompassing business architecture along with enterprise-wide IT architecture (EA = BA + EWITA)." Consequently, there are a number of definitions, which differ mainly by their scope. We understand the notion to cover business architecture as well and therefore later on in this work we refer with the term enterprise architecture to its broadest scope.

Nevertheless, Ross (2003) gives an excellent definition of *enterprise IT architecture* as "the organizing logic for application, data and infrastructure technologies, as captured in a set of policies and technical choices, intended to enable the firm's business strategy." Although devised for a somewhat narrower scope, we feel that it applies quite nicely for enterprise architecture as well. Furthermore, she defines the objectives of enterprise IT architecture to be identifying *IT capabilities*, which "specify what the architecture enables the business to do." These capabilities would include, for example, being able to access specific data for new applications, quickly add channels to existing processes, integrate data from related processes, ensure secure processing for elec-

tronic transactions, provide an extended online customer service or replicate systems in new locations (Ross, 2003). Again, we see the definition to apply well for the broader scope, too.

Enterprise architecture is the outcome, though an evolving one, of a strategic planning and management process where an *enterprise architecture framework* is applied to describe both the current ("as-is") and future ("to-be") states. These descriptions consist of high-level abstractions, or *architectural views*, of *system model* from different perspectives, or *architectural viewpoints* (Tang et al, 2004). However, the choice of viewpoints as well as the scope and precision of every view is specific to each framework. Additionally, some frameworks utilize a concept, which they designate as architectural segments, composite views or themes, to ensure that specific requirements, which cross-cut a number of architectural views, are coherently fulfilled. Those requirements are categorized as *non-functional requirements* and we term the concept in accordance with Rozanski and Woods (2005) as *architectural perspective*.

Commonly the ultimate strategic goal of enterprise architecture is said to be business/technology alignment and, therefore, as a general guide one ought to derive the architecture directly from business strategy and market dynamics. However, the present rate of change in the business environment and in business strategy itself as well as the diverseness of the strategy of an enterprise operating in multiple, global markets rarely offers IT a change to truly align itself. Ross (2005) argues that companies should instead select an operating model based on "(1) how standardized their business processes should be across operational units (business units, region, function, market segment) and (2) how integrated their business processes should be across those units." "In adopting an operating model a company benefits from a paradox: standardization leads to flexibility. By building a foundation of standardized technology, data and/or processes, our research shows a company achieves more business agility and responds to new market opportunities faster than its competitors." Consequently, by choosing one of the operating models business determines priorities for the development of its IT capabilities.

Business Process Integration	Low	High
High	***Coordination*** • Business unit control over business process design • Shared customer/supplier/product data • Consensus processes for designing IT infrastructure services; IT application decisions are made in business units	***Unification*** • High-level process owners design standardized process • Centrally mandated databases • IT decisions made centrally
Low	***Diversification*** • Business unit control over business process design • Few data standards across business units • Most IT decisions made within business units.	***Replication*** • Centralized (or federal) control over business process design • Standardized data definitions but data locally owned with some aggregation at corporate • Centrally mandated IT services

Business Process Standardization

Figure I-1. IT Characteristics of Four Operating Models (based on Ross, 2005)

As an organization's IT capabilities are unlikely to transition towards the target operating model unless it is actively pursuit after, it is vital that the guidelines set forth in its enterprise architecture are enforced in all IT investments. That, however, will not happen or at least it will be severely hampered, if the guidelines cannot be effectively and unambiguously communicated to all concerned parties. Therefore, besides concentrating on how we develop such architectures, we shall also work out how we document and represent them.

1.2 RESEARCH QUESTION

As a senior consultant at SysOpen Digia Plc the author has been involved in some assignments where client organizations' enterprise architectures have been audited or developed. A common theme among these assignments have been corporate level IT function's dissatisfaction to the ability of their enterprise architecture to steer IS (Information System) development projects. Yet another consistency for all of the cases was the presence of development outsourcing, competitive bidding, a large number of concurrent development projects, consolidation of IT infrastructure, and downsizing of the corporate level IT function in one form or another. These symptoms and trends are far from unique as they have been reported in many prior studies (see, for example, Mack and Frey, 2002, Ross and Westerman, 2003, Macehiter and Ward-Dutton, 2005, and Ranganathan and Jouppi, 2005).

In these assignments enterprise architecture documentation was identified as one core cause of the non-compliancy of projects and the excessive amount of project supervision needed. Even though the enterprise architecture itself contained right elements, the documentation failed to convoy them to application architects. Later, first as our own in-house methodology development and subsequently in a later assignment, these findings were further refined into an enterprise architecture documentation framework. The objectives we had for the refinement were basically the same as the research question of this thesis:

> *What is the optimal structure and form of enterprise architecture documentation for it to steer and enforce projects to stay in compliance with the architecture?*

The ability to steer has to do with proactively providing projects with relevant guidelines in order for them to stay in compliance with the existing enterprise architecture. Projects in this context include not only ongoing in-house ones, but also outsourced projects and those just in request for bids phase.

The ability to enforce is tightly related to steering and has to do with ensuring that projects are indeed in compliance with the existing enterprise architecture.

1.3 RESEARCH SCOPE

Although we touch the concept of enterprise architecture from various sides, the focus of this thesis is solely on documenting and presenting its essence – to paraphrase Ross (2003), the set of policies and technical choices concerning the organization of application, data and infrastructure technologies. Therefore, we will not cover in any notable depth the methodology and conventions, e.g. enterprise architecture framework, used in designing and maintaining enterprise architecture. Nor do we explore any of its management and governance issues. Consequently, while our aim is to design a universally applicable documentation practice that most enterprise architecture frameworks can be adapted to utilize, we acknowledge that the adaptation effort required is by no means uniform.

1.4 MAJOR CONTRIBUTIONS

The major contributions of this thesis are: (i) it augments a very scarce collection of research on enterprise architecture documentation and representation; and (ii) a novel documentation framework for it is introduced. When collecting background information, we were unable to find any prior studies on how enterprise architectures could and should be documented. A wealth of proprietary templates and guidelines do probably exist, but no academic studies with that focus. We try to make a contribution in that respect, although the context[1] in which this work has been carried out did not allow for a more thorough treatment of the subject. Therefore, partially we also try to open this field up for more research.

1.5 THESIS OUTLINE

The remaining chapters of this thesis fall into two main parts. Chapters II-III provide a detailed description of the problem domain, while Chapters IV-VI present the new method. Thus, the summary of related research work in Chapter II and the statement of the problem in Chapter III follow this chapter's introduction. Next, the proposed solution is examined in Chapter IV, and evaluated extensively in Chapter V. In Chapter VI we discuss the implications and limitations of the solution followed by a summary of the work with an outline of some possible directions for future research in Chapter VII.

[1] The bulk of the work has been done with and besides of a few rather short assignments.

CHAPTER II
RELATED WORK

> ***architecture*** *The set of design decisions about any system (or smaller component) that keeps its implementors and maintainers from exercising needless creativity.*
>
> *(D´Souza and Wills, 1998)*

In this chapter, we will examine existing work from academics and practitioners related to the documentation and representation of enterprise architectures. Prior work, although not in entirety directly related to our study, is heavily concentrated on enterprise architecture frameworks and thus we will first cursory go through the most significant ones in order to gain an insight into the structural conventions they employ. Architecture documentation is a vital part of software architectures, and even though the entities there are much more numerous and detailed, and the applied architecture views differ as well, we will next reference a few studies from that area. Finally, we cover architectural principles as a key property of enterprise architectures. We conclude with a summary.

2.1 ENTERPRISE ARCHITECTURE FRAMEWORKS

As already mentioned in the introduction, there are plenty of enterprise architecture frameworks to choose from. Tang et al (2004), Schekkerman (2004) and Allega (2005) list and cover in more detail the most established ones. However, due our scope of research we only focus on two aspects: (i) what conceptual structures are used and, (ii) what guidance is given on architecture documentation and presentation. Based on the former facet, the established enterprise architecture frameworks have been divided into two categories: matrix-shaped and tree-shaped ones.

2.1.1 MATRIX-SHAPED FRAMEWORKS

In 1987 Zachman proposed and a few years later (Sowa and Zachman, 1992) revised his well-known architecture framework, which is founded around a 5 x 6 matrix[1]. The matrix consists of five rows, representing unique perspectives of different stakeholders – *planner*, *owner*, *designer*, *builder*, and *subcontractor* – and six aspects or columns, identifying different types of information that are characterized by questions *what*, *how*, *where*, *who*, *when*, and *why*. As a result, each cell in the

matrix portrays certain aspect of enterprise architecture from a distinctive point of view (see Figures II-2). Consequently, the framework divides system model quite naturally into a set of sub-models.

	DATA *What*	FUNCTION *How*	NETWORK *Where*	PEOPLE *Who*	TIME *When*	MOTIVATION *Why*
SCOPE (CONTEXTUAL) *Planner*	List of Things Important to the Business	List of Processes the Business Performs	List of Locations in which the Business Operates	List of Organizations Important to the Business	List of Events Significant to the Business	List of Business Goals
ENTERPRISE MODEL (CONCEPTUAL) *Owner*	e.g. Semantic Model	e.g. Business Process Model	e.g. Business Logistics System	e.g. Work Flow Model	e.g. Master Schedule	e.g. Business Plan
SYSTEM MODEL (LOGICAL) *Designer*	e.g. Logical Data Model	e.g. Application Architecture	e.g. Distributed System Architecture	e.g. Human Interface Architecture	e.g. Processing Structure	e.g. Business Rule Model
TECHNOLOGY MODEL (PHYSICAL) *Builder*	e.g. Physical Data Model	e.g. System Design	e.g. Technology Architecture	e.g. Presentation Architecture	e.g. Control Structure	e.g. Rule Design
DETAILED REPRESENTATIONS (OUT-OF-CONTEXT) *Subcontractor*	e.g. Data Definition	e.g. Program	e.g. Network Architecture	e.g. Security Architecture	e.g. Timing Definition	e.g. Rule Specification
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK	e.g. ORGANIZATION	e.g. SCHEDULE	e.g. STRATEGY

Figure II-2. The Zachman Framework Matrix

Over the years the Zachman's framework has been widely adopted by the architecture community and it is incorporated into and influence of it can be found from many other enterprise architecture frameworks (Tang et al, 2004), e.g. Federal Enterprise Architecture Framework (*FEAF*) (CIO-Council, 1999), Treasury Enterprise Architecture Framework (*TEAF*) (Department of the Treasure CIO Council, 2000), and The Open Group Architecture Framework (*TOGAF*) (The Open Group, 2003). Schekkerman (2005) reports survey results that seem to indicate matrix-shaped frameworks being used in majority of enterprise architecture undertakings.

Although widely used, critique has also been brought out (Ross, 2003, Simsion, 2005, and Pulkkinen, 2006) indicating problems applying frameworks like Zachman's. Ross (2003) argues that because matrix-shaped frameworks treat all business processes, infrastructure, data and applications alike without any relative importance, it is hard to maintain focus in architecture activities. Thus the outcome has often been merely overwhelming volumes of detailed drawings that have little real value. She offers an approach where one focuses architecture efforts strictly on key business processes and evolves organization's IT architecture and architecture competency step-by-step through four stages. Each stage has a bit deeper and wider scope than the stage before it.

Pulkkinen (2006), on the other hand, finds Zachman's framework's perspectives problematic as they separate issues that in fact are deeply intertwined. Based on research of several enterprise

[1] In fact, the matrix has six rows, but as the last row represents the functioning entirety it is omitted.

architecture projects, Hirvonen and she (2004) suggest a matrix where there are four dimensions (business, information, application and technology architectures) and three abstraction levels (enterprise, domain and system scopes). A similar deviation from Zachman's original perspectives and aspects can be observed from many other matrix-shaped frameworks, for instance FEAF and TEAF.

Finally, according to Simsion's (2005) opinion Zachman's framework is "being advocated as the basis of an architected approach to enterprise-wide information systems planning and development, *on the basis of plausibility rather than evidence*." He has found very little evidence of success in applying it; instead, he argues, evidence of adoption is given.

In addition to having a common conceptual structure, these frameworks share obscure in the way their architecture views should be modeled, represented and documented. Very little, if any, guidance is given. The notable exception is TOGAF, which includes volumes of documentation, although relatively few of it is usable outright. Thus, another commonality is the need to adapt the framework for each case – selecting tools and developing modeling and documentation conventions.

2.1.2 TREE-SHAPED FRAMEWORKS

The consulting company Gartner has developed a proprietary enterprise architecture framework, including in it aspects from the framework by META Group[2]. The Gartner enterprise architecture framework (Naugher and Rosser, 2002) uses a radically different mindset than the matrix-shaped ones as it divides system model before the actual architecture activities into more homogeneous sub-models, typically using business process style as the factor. It also utilizes architectural patterns and bricks to bring even more structure.

Architectural patterns are used to represent a set of technology components, sometimes even multiple sets, which fulfill specific business process style related requirements. Thereby, architectural patterns further divide each business process style into a set of patterns applicable in it. Similarly, each architectural pattern references those bricks that are used to implement it. Bricks form the lowest, foundation layer of the Gartner enterprise architecture framework, and they document "specific instances of technologies, such as a database management system (DBMS) or client operating system" (Schulman, 2004) (see Figure II-3 on page 16).

[2] Gartner acquired META Group in 2005.

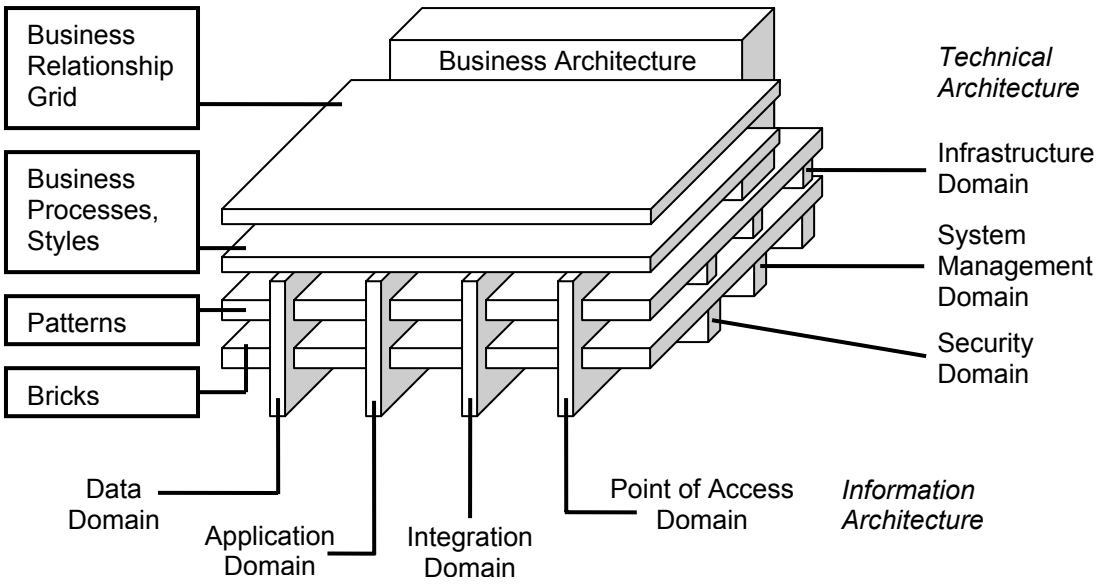


Figure II-3. The Gartner Enterprise Architecture Framework Model

The documentation of both patterns and bricks is subject to further structuring, too. Thus, the aspects of different information architecture domains – *data, application, integration* and *point of access domain* – and technology architecture domains – *infrastructure, system management* and *security domain* – concerns related to these parts are treated separately from each other. As the framework is proprietary, no concrete guidance of its use is publicly available. However, Robertson and Sribar (2002) shares some details of the conventions used in META Group's framework.

Another disadvantage of the framework being proprietary is that there are no independent studies and therefore no critique about it.

2.2 SOFTWARE ARCHITECTURE

Software architecture is a considerably more established discipline than enterprise architecture. The following definition is given in ANSI/IEEE 1471-2000 (IEEE, 2000): *The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.* Thus, where enterprise architecture is in general concerned of inter-system parts and aspects, software architecture focuses on intra-system parts and aspects.

One of the groundwork publications of software architecture is by Shaw and Garlan (1996) and the first formal standard in that area is ANSI/IEEE 1471-2000: Recommended Practice for Architecture Description of Software-Intensive Systems (IEEE, 2000). Architectural views have a central role in the standard, as is the case with a few other related works (Kruchten, 1995, Hofmeister et al, 2000, and Clements et al, 2003). The fundamental software architecture documentation principle from Clements et al (2003) summarizes quite well today's de facto approach – *documenting an architecture is a matter of documenting the relevant views and their relationships, and adding docu-*

mentation that applies to more than one view. It should be noted, however, that the understanding about the relevant views vary. Kruchten uses five fixed models, whereas Clements et al and the standard apply as many views as necessary based on system stakeholders and their concerns.

Software systems can be characterized by two principal properties: their (i) behavior and (ii) structure. Behavior determines how system acts and reacts to various stimuli, while structure establishes the different building blocks of which system is composed of and the exact way it is composed. Consequently, architectural viewpoints in software architecture are mainly either structural or behavioral in nature. Nevertheless, a system has often other qualities as well. These emergent properties, or non-functional requirements, are awkward if not impossible to represent in the same way as behavior or structure and, therefore, they are usually formularized as natural language statements. Non-functional requirements include constraints and qualities – both development-time and run-time qualities (Malan and Bredemeyer, 2001) – and together with architectural views and additional documentation they form the architectural description, or documentation, of a system.

Design patterns were introduced by Gamma et al (1994) and since then patterns have become an essential instrument in documenting and sharing common software design solutions to recurring problems. Today patterns have permeated through to many other areas as well, for instance system analysis (Fowler, 1997), enterprise applications (Fowler, 2003), and, of course, enterprise architecture. The basic form of a pattern varies slightly, but typically you can expect to find from its description (i) its name, a description of (ii) the problem it intents to solve, (iii) when to use it, and (iv) the solution, and possibly (v) examples, known uses, or related patterns. Thus, patterns are "just" descriptions, although of practice-proven solutions to recurring problems, and hence one must adapt it to the problem and context at hand. As such, they are also a great way to document something just once – something that repeats in almost the same form multiple times.

2.3 ARCHITECTURAL PRINCIPLES

Architectural principles are non-functional requirements. Bredemeyer (2001) defines them as "statements of preferred architectural direction or practice." Thus, in the context of enterprise architecture, they state the properties and qualities that the architectures of new information systems shall possess. Some of the enterprise architecture frameworks employ architectural principles as a key property of their architectural views. These frameworks include Gartner, TEAF, and TOGAF. However, the use of architectural principles is typically limited to the about dozen core ones. Lindström (2006) has studied the usage of architectural principles in more detail.

TOGAF uses architectural principles to define the underlying general rules and guidelines for the use and deployment of all IT resources and assets across the enterprise (The Open Group, 2003). It encourages using a standard format for defining principles. The recommended template con-

tains besides (i) a short name and (ii) definition statement (iii) the associated rationale and (iv) implications statements. The definitions are stated in natural language, which is "by its nature highly dependent upon assumption and definition (even disregarding ambiguity)." (Mannion and Keepence, 1995). Therefore, it is vital that enough attention is paid to the correctness, completeness and consistency of the statements. Mannion and Keepence (1995) suggest a simple technique, which could help enterprise architects to improve the quality of architectural principles. SMART requirements, or principles, have the following characteristics:

***S**pecific*: It is clear, without ambiguity, yet simple and its terminology has been consistently used throughout the document.

***M**easurable*: Its fulfillment can be verified.

***A**ttainable*: It is technically feasible.

***R**ealizable*: It is achievable with the given resources and constraints.

***T**raceable*: It is possible to trace its business justification.

2.4 SUMMARY

In this chapter, we have examined the existing work related to enterprise architecture documentation and representation. The most established enterprise architecture frameworks studied where found to apply either a matrix-shaped or tree-shaped structure as their inner-organization. This has major implications to the structure of documentation needed to represent them. Unfortunately, we also found that no framework is particularly strong in guidelining documentation – the majority of them actually being very weak.

In addition, the software architecture discipline was found to make use of architecture views in documentation, and finally, some of the enterprise architecture frameworks were found to incorporate architectural principles as one key element.

CHAPTER III
PROBLEM STATEMENT

> ***enterprise architecture*** *An enterprise is made up of many interacting systems of various kinds. Enterprise Architecture identifies these systems, their key properties, and their interrelationships, and plans for and guides the evolution of the enterprise systems to support and enable the evolution of the enterprise in its pursuit of strategic advantage.*
>
> *(Ruth Malan, Bredemeyer Consulting)*

In the previous chapter, we gave an overview of the existing work related to the problem area this study focuses on. Now in this chapter, we will elaborate on the specific problems that were identified and the background behind them. We start with giving an overview of two case projects, which exemplify the problems. Next, we formulate the problem statement, and conclude with a summary.

3.1 CASE 1

This case customer is a Finnish group, which has a considerable number of rather independent affiliates. All of their core information systems are developed and operated at the group level, but each affiliate is also able to make independent IT investments. IT administration and architecture supervision are at the group level, whereas all development and operation have been outsourced to a joint venture. The customer's IT administration had some concerns about the present way of documenting their enterprise architecture, and before launching a major implementation project of a new technical architecture, they conducted an audit of it. Based on its findings, they started a project to develop their enterprise architecture documentation conventions.

The key finding of the audit were, that the documentation would have been unable to steer or enforce future projects in any meaningful way. The reason behind that was the burial of all key architectural decisions underneath volumes of explanatory text. Furthermore, many decisions were written in a very obscure way, as the documentation covered all platforms at once. In total the documentation contained hundreds of pages of which almost all treated issues like the ACID properties of transaction handling or UML modeling. It simply tried to teach all key aspects of software architecture to the reader. Another problem was the fact that it contained four or five

parts, each written by a different author. Quite naturally they all looked very different, which made reading and comprehending it much harder.

The reason for the failure was the absence of documentation guidelines of any kind from the enterprise architecture framework they used. Therefore, each author had used the documentation conventions they were familiar with, e.g. software architecture documentation conventions.

3.2 CASE 2

This case customer is a global corporation, which has a number of divisions. Its IT organization varies by division from ones that have a considerable IT function of their own and which thereby are rather independent to ones that rely completely on the corporate level IT services. Although the customer has some in-house development both at the division as well as at the corporate level, it outsources heavily. The customer was at the corporate level implementing a new consolidated data center and the first platform services for applications deployed there. Therefore IT administration needed enterprise architecture-level documentation to distribute to their subcontractors as well as their own developers. The problem they faced was that the established enterprise architecture frameworks did not really support, or at least they offered no guidelines for, the division of the documentation to company internal parts and distributable parts. Another problem aroused from the fact that the shared data center does not give much of a choice of technologies to implementation projects. Instead, there was an express need to restrict some product features from being used to ensure the operability of the data center. This proved to be a notable omission in current enterprise architecture frameworks – they still concentrate on technology selection, which is loosing importance due to the growth of consolidated data centers and commoditization of both hardware and software, whereas feature selection, e.g. selectively allowing or disallowing the use of different product features, is quickly getting more and more important. Yet another problem, common with the first case, was the number of domain experts that were going to take part in "writing" the architecture documentation.

3.3 SYNTHESIS

These two cases illustrate the barriers faced today by corporations trying to introduce enterprise architecture for themselves. We cannot even argue that this happened because of too ambitious scope, as both of the cases were small starts. Our view is that what happened was partly due to the mismatch of current enterprise architecture documentation conventions with the reality where corporations try to produce the documentation or utilize it. Clearly, there seems to be an over-emphasis on "as-is" architecture and, consequently, de-emphasis on "to-be" architecture and characteristics present in enterprises today.

To synthesize and elaborate, the obstacles found are:

ISSUE 1 – EXTENSIVE ADAPTATION NEEDED

The established enterprise architecture frameworks do not provide enough guidelines or templates for writing architecture documentation – too large cap is left to close for those who adapt the framework.

ISSUE 2 – LACK OF CONTEXT

The matrix-shaped frameworks tend to represent architecture model in a way that easily leads one to document each architecture view without further substructure. In particular with application and technology architectures this means you have to treat matters "out-of-scope". For example, when you mix the architectural issues of mainframes, midrange and workgroup servers into same document, it is remarkably difficult to make it readable and practical. Additionally, the issues are easily treated in such high level that only the true commonalities get documented properly.

ISSUE 3 – CROSS-CUTTING ASPECTS

The "cross-cutting" aspects of enterprise architecture, like security, tend to spread out to different parts of the documentation. This is a nuisance when a group of domain experts try to produce the documentation.

ISSUE 4 – PART-WISE DELIVERY

The established enterprise architecture frameworks do not, by default, structure documentation in such a way that parts of it could be distributed as-is to subcontractors.

ISSUE 5 – TECHNOLOGY FEATURE SELECTION

The established frameworks do not support technology feature selection. They focus on technology selection without touching where it can be used and where not, or what features of it can be used and what not. For example, contemporary databases can typically be run on workstations and even mobile terminals and one can implement a file, ftp or web server with it, even application server features are nowadays built-in. Surely one has to make quite a few decisions more besides selecting one brand.

3.4 SUMMARY

In this chapter, we first described the problems and obstacles found during two case assignments. The descriptions hopefully clearly depict how current enterprise architecture documentation conventions fall short in many ways. Next, we synthesized and elaborated those problems and obstacles to a problem statement.

CHAPTER IV
PROPOSED FRAMEWORK

> ***enterprise IT architecture*** *The organizing logic for applications, data and infrastructure technologies, as captured in a set of policies and technical choices, intended to enable the firm's business strategy.*
>
> *(Ross, 2003)*

In this chapter, we will describe the proposed solution that strives to address those issues which we described in the previous chapter. Basically the proposed solution is a documentation framework. First, we will specify the objectives it pursues to achieve. They are essential to understand the choices and tradeoffs made. After that, we will introduce the concepts and terms that are used throughout the framework. Next, we will share some of the reasoning behind our framework followed by the definition of basic architectural elements on which the framework builds. After enough background has been covered, we will present the whole framework – its structure and the documentation templates for each part. That will give an overview of what a full documentation of enterprise architecture might look like. Finally, we conclude with a summary.

4.1 OBJECTIVES AND PRINCIPLES

The proposed solution we are about to introduce is substantially different from many respects compared to the established enterprise architecture frameworks. The reason, in our opinion, is a different set of objectives that this framework tries to achieve. While the objectives of most of the other frameworks are not clearly stated and, thus, are unavailable for comparison, we state ours. Derived from them are a set of principles that are followed throughout the framework.

The objectives we had when designing the proposed solution were:

1. The framework shall be readily usable without adapting.
2. The framework shall be structured enough to provide a clear context for exposition, but not too structured to overwhelm the reader.
3. The framework shall be able to convoy all pivotal decisions and guidelines unambiguously.

4. The framework's structure shall enable piecewise delivery of the documentation without revealing company internals.

The objectives gave rise to the following principles:

1. There should be a ready-to-use document template for each piece of documentation, if feasible. Avoid all lists-of-things-to-include-in-the-documentation.
2. Favor architecture principles; minimize the use of free text and diagrams.
3. Strip any such content that does not contribute to the convoy of pivotal decisions and guidelines.
4. Keep the logical solution descriptions and physical technology and security descriptions in separate pieces of documentation; the same applies for logical solution descriptions and any higher-level descriptions, too.

4.2 CONCEPTS AND TERMS

First of all, we use four views or dimensions to the enterprise architecture. Those dimensions are:

1. Business architecture
2. Solution architecture
3. Platform architecture
4. Security architecture

The choice of dimensions deviates slightly from what is conventional today. The business architecture view is used to illustrate the interconnections between business processes and information systems. Fundamentally, its purpose aligns well with its intent in other enterprise architecture frameworks, albeit its content is deliberately stripped-down – we do not see the point of including aspects like organization structure, business functions and services, or business roles as they are neither created by enterprise architects nor needed to convoy the pivotal decisions.

The solution architecture view, as a matter of fact, combines several views present as separate entities in other frameworks into one. These views are typically known as data or information architecture, application or systems architecture, and integration architecture. It is composed of purely logical descriptions, which reference those "components" from physical platform and security architecture views that are needed to "implement" it.

The platform architecture view is also known as infrastructure or technology architecture in some enterprise architecture frameworks. It describes the technology components and shared services of all physical runtime platforms.

Finally, the security architecture view consists of descriptions of security zones. Security zones are physical areas sharing the same security conventions and rules, related for instance to physical access, network access, and confidentiality-level of data stored and processed.

The following terms are used for individual parts of documentation, or artifacts:

- The business architecture view consists of *usage charts*.
- The solution architecture view consists of *architectural templates* organized according to their *architectural style*.
- The platform architecture view consists of *application foundations* and *network connections*.
- The security architecture view consists of *security zones*.

With the term *architectural elements* we refer to application foundations, network connections and security zones. Although the framework builds upon and, thereby, more or less mandates these four dimensions, it does not prevent one from incorporating other relevant dimensions as well. However, unless

4.3 RATIONALE

Enterprise architecture is an instrument for the strategic management of enterprise's IT assets and resources. It is **not** a repository of models and architectural information. Strategy, on the other hand, is about pivotal, overarching decisions. Therefore, first and foremost, enterprise architecture must be a vehicle to convoy those decisions. That is the reasoning behind favoring architectural principles and minimizing the use of models and free-form text. Models just are not able to unambiguously communicate such decisions and writing them in free-form takes proficiency. As illustrated by our first case study, engineers who are proficient enough to write down those decisions in free-form are rare. Instead, writing them down as architectural principles in a narrow, clear context is much more comfortable.

Building this narrow, clear context is the rationale to "componentize" the documentation. Otherwise, it would have been necessary to build this context by structuring, e.g. using chapters and subchapters, bigger documents. However, as the entities that need to be documented vary greatly it would have been hard, if not impossible, to develop document template for that. Essentially, we would have ended up where most enterprise architecture frameworks are today – to guideline documentation using lists-of-things-to-include-in-the-documentation.

Componentization brings about other benefits as well. One is that you can quite selectively disclose only parts of the documentation as independent subsets, which still make sense although they might miss some detailed information. One example could be that in call for bids phase you deliver just one or more architectural templates without the descriptions of the used architectural

elements to all potential bidders. The bidders still get a fairly good picture of the target architecture, even though they do not have all the technical details. Later on, when the bidders are short listed, you can disclose the architectural element descriptions to them for them to get a full picture of the rules they must play with.

4.4 ARCHITECTURAL STYLES AND TEMPLATES

Architectural templates define elementary ways of using architectural elements to build applications for specific purposes. As such, they are purely logical definitions, which reference the descriptions of those architectural elements that are necessary for its operation. Software architects are mostly interested about these as they define the bodyworks of compliant architectures on which architects can design and build their own applications. All architectural templates have a name, which is used to identify and reference it.

Architectural templates are known as architecture patterns in some other enterprise architecture frameworks. However, the choice of different term is a deliberate one, as patterns, by their definition, always require adaptation – the architecture defined by an architectural template, on the other hand, is by itself already a working one and therefore it is not necessary to adapt it. Nevertheless, larger solutions typically embody several architectural templates, whereupon one commonly needs to adapt them to fit together.

As large corporations might document a large number of different architectural templates, the need to organize them arises quickly. The style of architecture is used in the framework to group and organize its architectural templates. Style, of course, is a subjective quality, but here one can make use of the application area, target platform, or development technology as template's style. Figures IV-4 and IV-5 give an example of architectural styles and templates.

ARCHITECTURAL TEMPLATES

Architectural templates define elementary ways of using the elements of the shared infrastructure to build applications for specific purposes. As such they are the highest-level descriptions of the architecture. The detailed descriptions of elements named in architectural templates can be found later on under subsequent sections.

Online Transaction Processing	
Template	Description
· Smart Client OLTP Architectural Template	Generic OLTP application architecture based on .NET-clients and Web Services
· Terminal Client OLTP Architectural Template	High-volume OLTP application architecture based on 3270-terminals
· Web Client OLTP Architectural Template	Generic OLTP application architecture based on web browsers and J2EE middle-tier
Workgroup Applications	
Template	Description
· ...	
Reporting	
Template	Description
· ...	

Figure IV-4. Architectural styles and templates

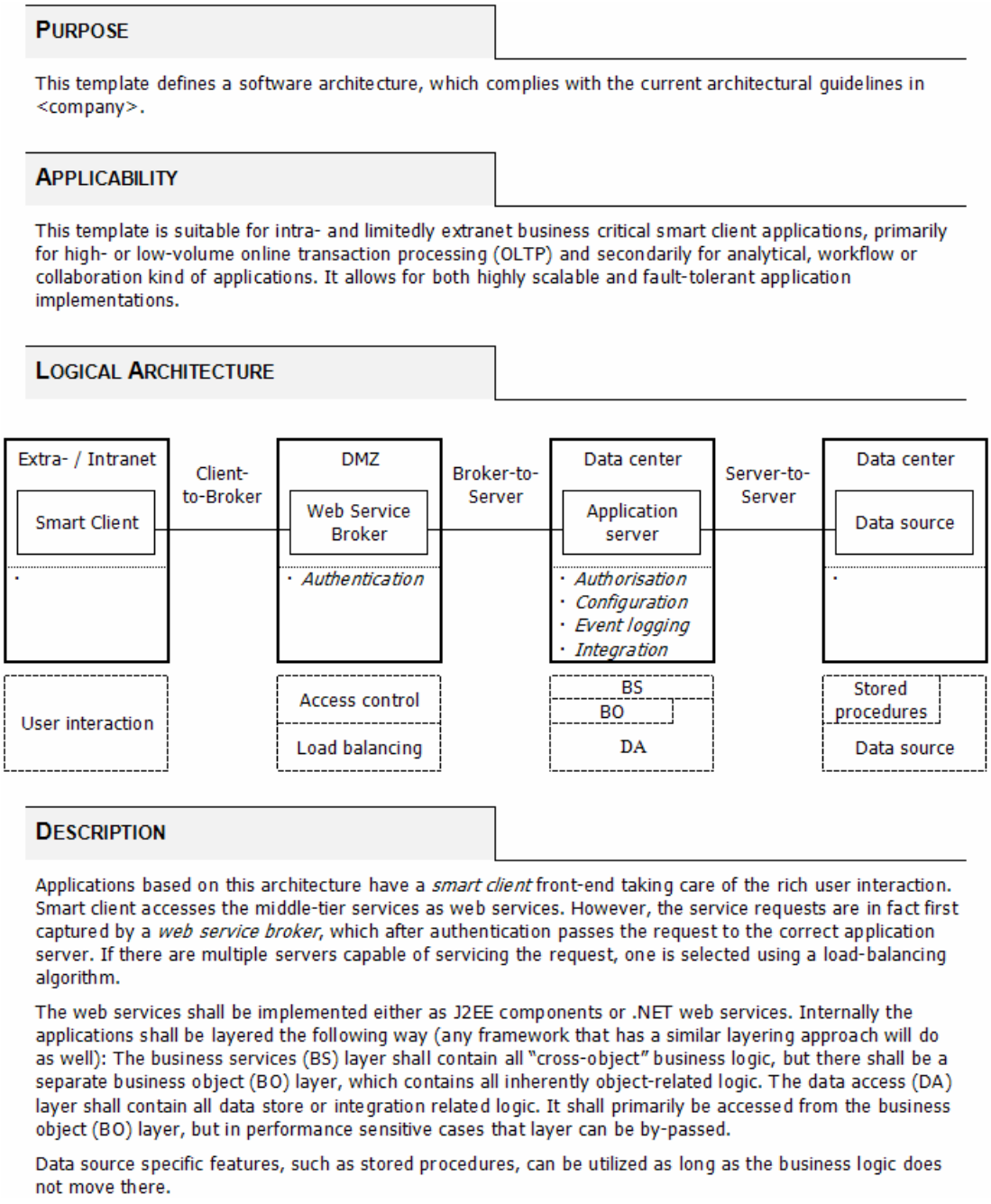
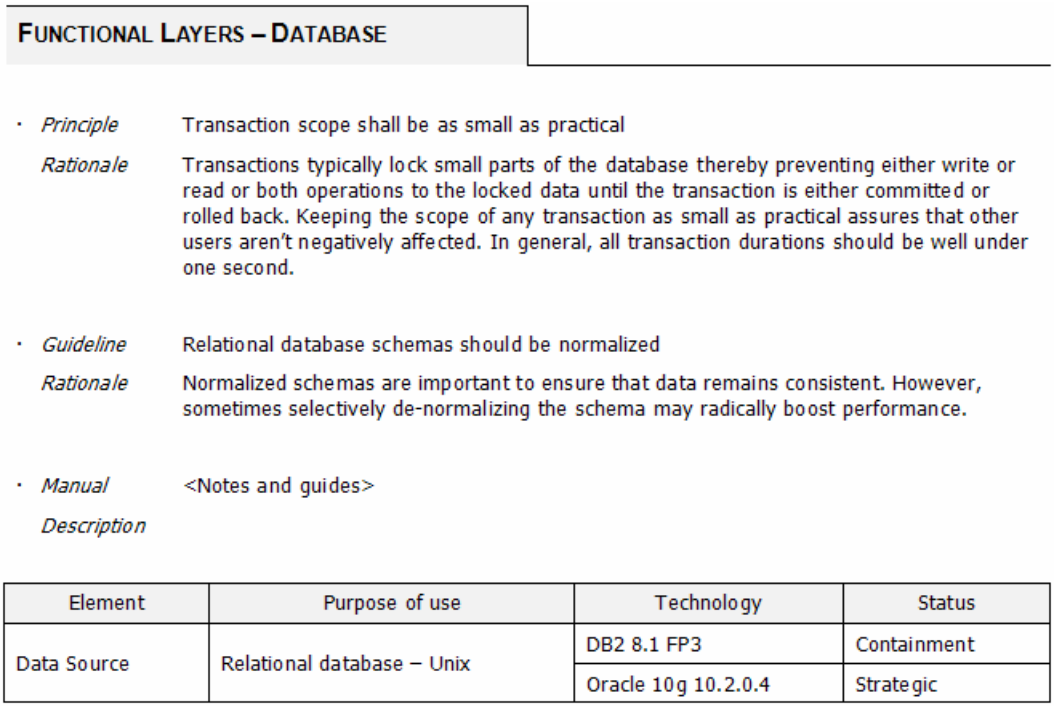


Figure IV-5. Architectural template (first page)

4.5 ARCHITECTURAL ELEMENTS – APPLICATION FOUNDATIONS, NETWORK CONNECTIONS AND SECURITY ZONES

Architectural elements are those that architectural templates are made up of. Architectural templates are logical designs or blueprints, whereas architectural elements have a physical counterpart, they are rich in principles and they have a technology roadmap.

Application foundations are descriptions of runtime platforms on which parts of application's code execute. Examples of such are application servers, database servers, transaction monitors, browsers and workstations. Their documentation uses a layered approach, where layer by layer the principles and technology roadmap are described as can be noted from Figure IV-6. However, as the runtime platforms differ radically from each other, the documentation is adapted by suppressing unnecessary layers. All application foundations, as well as other architectural elements, have a name, which is used to identify and reference it.



FUNCTIONAL LAYERS – DATABASE

- *Principle* Transaction scope shall be as small as practical

  Rationale Transactions typically lock small parts of the database thereby preventing either write or read or both operations to the locked data until the transaction is either committed or rolled back. Keeping the scope of any transaction as small as practical assures that other users aren't negatively affected. In general, all transaction durations should be well under one second.

- *Guideline* Relational database schemas should be normalized

  Rationale Normalized schemas are important to ensure that data remains consistent. However, sometimes selectively de-normalizing the schema may radically boost performance.

- *Manual* <Notes and guides>

  Description

Element	Purpose of use	Technology	Status
Data Source	Relational database – Unix	DB2 8.1 FP3	Containment
		Oracle 10g 10.2.0.4	Strategic

Figure IV-6. Documentation of an architectural layer

Network connections are wired or wireless connections linking two application foundations together. Security zones are physical areas where application foundations are located. Examples of such are demilitarized zone, data centers, internet, intranet and extranet. Again, the documentation for both uses a similar layered approach, although with only one layer.

4.6 USAGE CHART

Architectural templates are designed to support certain IT capabilities and new information systems built in accordance with the templates most likely support them as well. However, enterprises typically have a huge variety of information systems of different age and technology-base. Therefore it is essential that enterprise architecture includes a tool, which can give an overall status of how well a certain capability is supported by present information systems. Usage chart is such a tool.

Information systems' purpose is to support business and its processes and as the great majority of systems' changes originate from changes in those processes, the only viable option is to chart usage by business processes. However, charting just usage does not reveal much. Therefore the chart is augmented with the aspects that are relevant for the sought-after IT capabilities. Hence, if business requires IT to be ready to quickly add channels to existing processes, the usage chart might be augmented with information like client type used, volume of activity supported, average response time, and programmatic interface type in the case the same functionality is available as services. An exemplar of a usage chart of a claim processing business process augmented with IT capability related information is depicted in Figure IV-7.

BUSINESS PROCESS

Task	Information System	Reason	Client		
			Win	Web	3270
Identify customer	Customer Register	Identify, check personal information, note any trouble entries			X
	Insurance Register	Validate insurance		X	X
Book claim	Compensation System	Book customer's claim		X	
Book decision	Compensation System	Book resolution to customer's claim		X	
Confirm payment	Payment System	Accept payment for pay off			X

Figure IV-7. Information system usage chart of a business process

As anyone can quickly observe, adding for instance a web-based self-service channel to the business process might not be entirely straightforward. This kind of feedback is very valuable when sketching business process changes and, reversely, the charts can be inspected to detect problem areas of existing information systems, which might hinder or prevent such changes. Consequently, by prioritizing its processes and stating the desired operating model for IT business can directly influence IT's strategic agenda.

4.7 DOCUMENTATION STRUCTURE

The overall structure of the documentation is illustrated in Figure IV-8. It should be noted that architectural templates share the descriptions of architectural elements and hence the elements do not grow in number that fast.

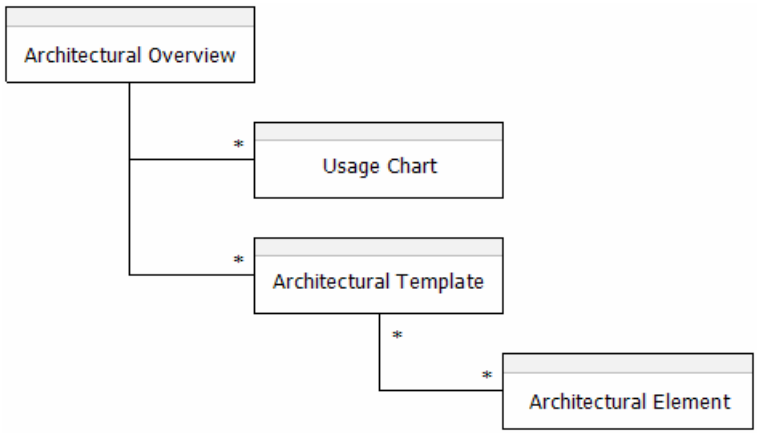


Figure IV-8. Documentation overall structure

4.8 SUMMARY

In this chapter, we introduced our proposed solution by describing the objectives and principles behind it, the concepts and terms, and finally the rationale for some of the choices and tradeoffs made. In the last part we explained the entities to document, the document templates available and the overall structure of the documentation.

CHAPTER V
EVALUATION

> ***software architecture*** *Architecture is the organizational structure of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems.*
>
> *(UML 1.3)*

This chapter zeros in on evaluating the solution presented in the previous chapter against the issues discussed in the chapter before that. Evaluation is performed against the issues one-by-one and in the end some preliminary results based on user interviews are given in addition to summarizing the evaluations.

5.1 ISSUE 1 – EXTENSIVE ADAPTATION NEEDED

Not providing documentation guidelines or templates forces users to develop them, typically costly through trial-and-error. To avoid that, the proposed documentation framework contains a document template for each and every entity. The templates can readily be taken into use, as they contain all boilerplate text. No further adaptation is needed. To support limited scale of extension, the templates also embody a number of extension points where links to custom documentation, for instance additional architecture views, can be attached.

5.2 ISSUE 2 – LACK OF CONTEXT

Lack of context easily leads to ambiguous and confused description. To fight that, the proposed documentation framework has a distinct structure, parallel to the structure of real world information systems. The structure manifests itself both through the documentation parts, artifacts, that the conceptual model of enterprise architecture is divided up into as well as in the layers that the description of each part is partitioned into. The outcome is a natural and crisp context for stating the guiding principles of enterprise architecture. The following example demonstrates this.

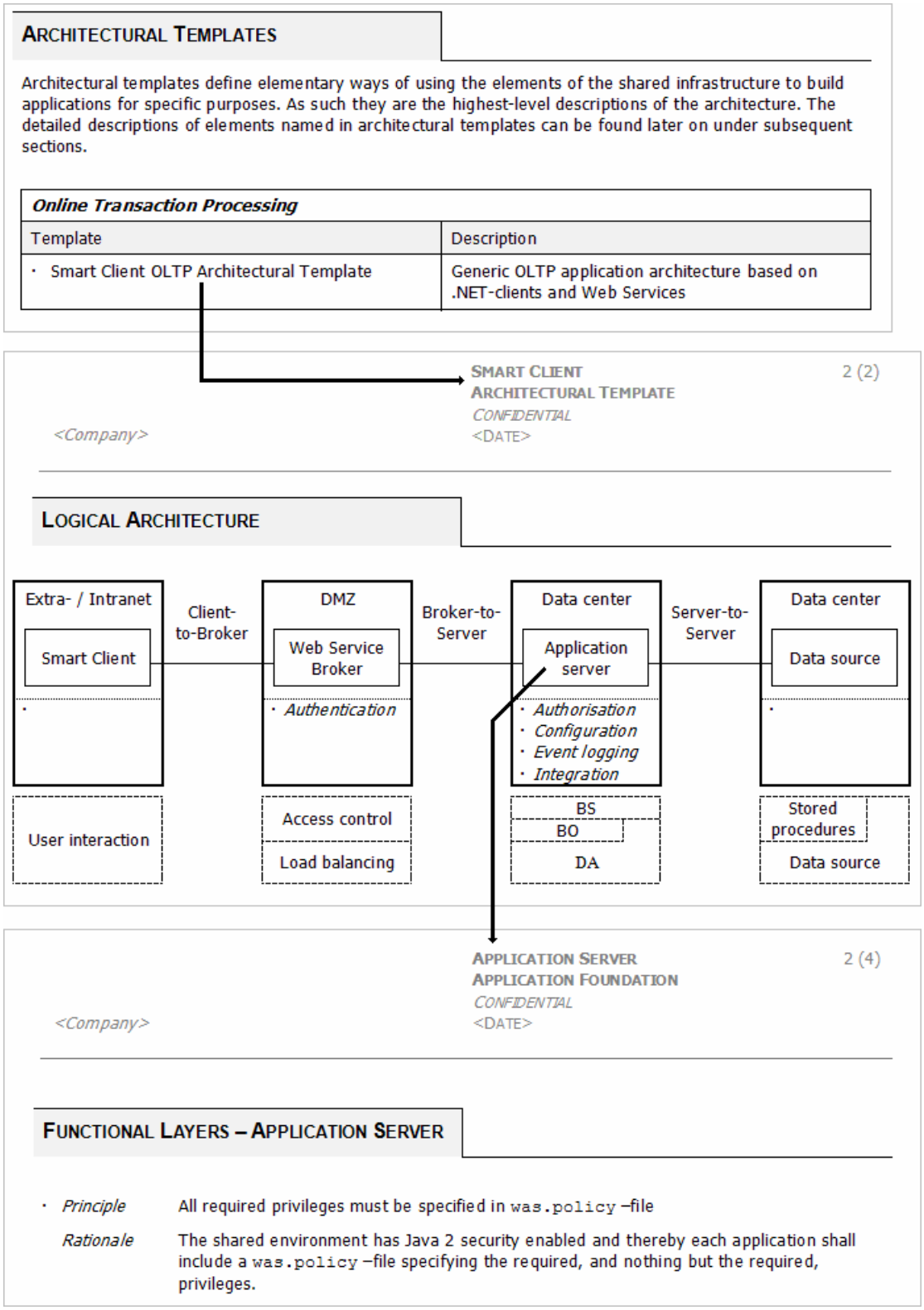


Figure V-9. Context building mechanism

5.3 ISSUE 3 – CROSS-CUTTING ASPECTS

Dividing enterprise architecture for instance into four domains – business, information, application and technology architectures – creates a circumstance where some aspects of enterprise architecture, notably security and system management, need to be discusses in almost each of those domains. For example, what information can be stored and processed where in information architecture, authentication and authorization details in application architecture, and details of firewalls and other security appliances in technology architecture. Having two or more experts, one principal author and several assisting ones, writing one document obscures ownership and responsibilities as well as introduces a certain amount of inertia to the process. The proposed documentation framework approaches the issue in a couple of ways. First of all, security architecture in the framework is a domain on its own. Framework's structure is specifically designed to allow that. In fact, security zones are an exception in that sense, as other architectural elements contain matters from various domains. The matters, however, are partitioned to elements' layers so that they do not cross multiple layers. Second, using architectural principles as the fundamental description form promotes group work where principles are suggested, discussed, edited and finally settled. Thereby ownership, responsibility and work distribution issues tend to diminish substantially.

5.4 ISSUE 4 – PART-WISE DELIVERY

Subcontractors have a distinct need for enterprise architecture documentation, already when negotiating about a project, and not being able to provide it for them early enough, typically due its confidentiality and the applied security policy, affects their offer negatively – for instance higher risk margin and work estimates. The proposed documentation framework partitions that subset of documentation, which is mainly of interest to software architects, into logical designs, aka architectural templates, and descriptions more related to the physical runtime environment of information systems, aka architectural elements. By their nature architectural templates include very little details of the actual runtime environment. Therefore they can be regarded as less confidential and in many cases disclosed to subcontractors even without non-disclosure agreements. Figure IV-5 (see page 27) demonstrated an architectural template.

5.5 ISSUE 5 – TECHNOLOGY FEATURE SELECTION

Unless regulated accepted technologies and products could get used in unforeseen ways. There is nothing wrong about devising new uses for existing IT assets, except that might render the neighbor applications inoperative. Commonly the focus is still on technology/product selection, although feature selection is nowadays more important. The proposed framework especially supports it by having multiple layers on top of the technology/product layer. It is convenient in these layers to take position on matters like what protocols can be used for communication and which application programming interfaces (APIs) are permitted. As many of these areas keep changing steadily, a current option might be replaced and thus be prohibited tomorrow. Therefore architectural elements have a roadmap on every documentation layer. The following example clarifies this whole concept.

FUNCTIONAL LAYERS – DATABASE

- *Principle* Use JMS queues for asynchronous messaging
  Rationale Using transactional message queue can simplify the architecture of applications having to deal with asynchronous services or processing. Use the JMS API to access such queues.

- *Principle* Use XML queries with discretion
  Rationale Large result sets from XML queries can tax database resources, especially memory, extensively. XML queries shall therefore be limited to cases where a modest number, less than 50, rows are expected. Use application server to turn result sets larger than that to XML.

- *Principle* Other database extensions are prohibited
  Rationale Database products carry a tremendous number of features. However, thoroughly testing these features in a shared environment is an unfeasible task and therefore other extensions are by default prohibited. Consult with the EA office, if an extension is badly needed.

- *Manual* Oracle Database 10g Release 2 Documentation Library
  Description Full documentation library for Oracle database 10g.

Element	Purpose of use	Technology	Status
Data Source	Relational database – Unix	DB2 8.1 FP3	Containment
		Oracle 10g 10.2.0.1	Strategic
Oracle 10g	.NET Data Provider	Oracle Data Provider for .NET 10.2.0.2	Tactical
Oracle 10g	JDBC driver	DataDirect Connect for JDBC 3.6	Tactical
Oracle 10g	AQ programmatic interface	JMS 1.1	Strategic

Figure V-10. Database feature selection

5.6 SUMMARY

Thus far we have formulated a problem statement based on the findings from two case projects in chapter 3. Next we presented the proposed enterprise architecture documentation and representation framework in chapter 4. In this chapter we evaluated this proposed framework against the five issues found in the problem statement. First, we noted that as the proposed framework contains readily useable document templates, it does not suffer from a large adaptation effort. Second, we were shown that its structure supports well the formation of context, which is important for description's quality. Third, the cross-cutting aspects issue was demonstrated as solved, again by its structure. Fourth, we have seen how parts of the documentation are in such a high logical level that they can quite well be disclosed to subcontractors as-is, thus solving the issue. Fifth, document layers' ability to capture feature selections was demonstrated solving the last issue.

Consequently, it has been shown that the proposed documentation framework is indeed designed in such a way that the issues identified in problem statement are addressed. This claim is partially supported by an informal interview of two architect-level developers from a subcontractor working for the second case customer. To summarize the content of the interview briefly, they felt that the architecture documentation delivered to them gave a good picture of the target architecture and helped them to create the software architecture for their solution. The main critique they had was about the structure of the framework as they felt that it took some time to be acquainted with it to find ones way around.

CHAPTER VI
DISCUSSION

> ***software architecture*** *Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled.*
>
> *(Eoin Woods)*

In the introduction of this thesis a number of challenges concerning present enterprise architecture documentation and representation conventions were identified. In particular, the ability to steer IS projects during design and development and the ability to enforce their outcomes to be in compliance with the architecture were found suboptimal. In this chapter we will discuss how the research question in the introduction has been addressed by the proposed documentation framework.

6.1 RESEARCH QUESTION

As a reminder, we formulated our research question as:

> *What is the optimal structure and form of enterprise architecture documentation for it to steer and enforce projects to stay in compliance with the architecture?*

We will answer it in two parts. First, what qualities of the proposed framework prompt it to steer, and secondly, what qualities prompt it enforce project to stay in compliance with the architecture.

6.1.1 ABILITY TO STEER

Steering projects involves providing them with the preferred architectural directions and practices. If one fails to do that, perhaps because the target has not been clearly settled or explicitly documented, then steering quickly turns into personal participation. However, with dozens or hundreds concurrent IS projects that corporations like those in our case projects typically run nowadays, personal guidance is an unsustainable approach. Inadequate steering causes architectural decision making to creep into project level where choices are more or less ad hoc. Kähkipuro (2005) calls this "the concealed architecture".

CLEAR STRUCTURE

However, merely having the target architecture documented and on hand for projects is not enough. Software architects prefer quick and easy access to all relevant rules. Therefore the structure of architecture documentation is of paramount importance. Making them wade through volumes of documentation to locate the appropriate pieces is a bad sign. Doing so, architects are likely to overlook some details even if they allocate time to bury in it. More typically, though, they simply ignore it.

We believe that it is more natural for architects to approach enterprise architecture documentation through application areas, e.g. browser-based online transaction processing or client-based analytical processing, than through orthogonal architectural domains or views, e.g. applications or information. Even if the documentation of each architectural domain is structured around application areas, one would still need to "collect" the relevant pieces from separate parts of the documentation instead of having them readily in just one.

Still, our proposed documentation framework, as presented in previous chapters, does not highlight the concept of application area. In fact, it is structured around architectural templates. Templates are application area specific constructs that in addition to a logical representation of a solution include an outline of those technologies that are used to implement it. The technological details can be found one level deeper, from the documentation of architectural elements. It would have been possible to make the documentation of architectural templates self-contained. However, that would have replicated large parts of the lowest-level documentation in many templates forming a maintenance nightmare. In our opinion, the proposed three-level structure, e.g. overview – architectural templates – architectural elements, is the simplest feasible. Clearly, it cannot be made any simpler without severe consequences, and no real reason has been found to make it more complex.

Nevertheless, this simple and pragmatic structure supports software architects to quickly locate the pertinent parts of enterprise architecture documentation, parts that contain all relevant rules – no more or no less. Furthermore, this structure has many additional advantages as described in the previous chapter.

UNAMBIGUOUS CONTENT

Besides structure, content is equally, if not more, significant. Enterprise architecture is more about the properties and qualities expected from the software architectures of information systems than it is about information systems, their properties and interrelationships. Thus, enterprise architecture is first of all a meta-architecture, which places non-functional requirements on

the architectures of to-be-built information systems. We do not model non-functional requirements of software systems and therefore we should not model non-functional requirements of architectures. Despite the fact, modeling and documenting models has thus far been the common norm in enterprise architecture. The challenge with models and modeling is that (i) capturing the time aspect is difficult, (ii) performing it at the appropriate level of detail is tricky, and (iii) as models are abstractions where you selectively retain some details and omit others, they are always subjective, ambiguous views. Hence, models in enterprise architecture typically overemphasize the "as-is" state providing too much detail, while they understate the "to-be" state. That is quite understandable as detailed modeling of aspects of information systems not yet build is about the same as half-planning them, unfeasible.

The approach we chose for the proposed framework focuses on representing the non-functional requirements. It does that by using architectural principles and consciously avoiding any ambiguous forms of documentation, like models and natural language. This helps keeping the documentation short, which again makes it easier and more likely for software architects to actually use it. Furthermore, the quality of the documentation can be improved and controlled by utilizing the SMART technique both when writing and accepting it.

TECHNOLOGY FEATURE SELECTION

Most of the technologies enterprises currently employ are mature ones and thus the respective products they have chosen typically contain an excessive number of additional features. Good examples are application servers, databases, and web servers. The risk that relates to these additional features is that even though the product itself is mature, a feature might be immature and it might have some surprising restrictions. In particular in consolidated runtime environments, uncontrolled use of such features might cause far-reaching ill effects.

The challenge with technology feature selection is that these features facilitate building systems and unless you place some rules on their use software architects will take advantage of them. Typically enterprise architecture frameworks focus on technology selection forgetting to rule what features of it are allowed. As a result, there might be a couple of servers more in the data center as a new information system, for some reason, did not run well with the others on a shared server. By stating the restrictions up front, software architects can be steered to select such a feature set that projects do not end up in costly surprises when about to go live.

The proposed documentation framework uses a layered approach, as described in previous chapters, which makes it natural to describe, for instance, how databases are allowed to be integrated to other systems. Both allowed features and allowed programmatic means, e.g. application programming interfaces (APIs), to access the features can be described.

Using these three qualities, e.g. clear structure, unambiguous content, and technology feature selection, we have thus answered the first part of our research question. The proposed enterprise architecture documentation and representation framework has the optimal structure and form for it to steer IS projects.

6.1.2 ABILITY TO ENFORCE

If IS projects are steered the greatest part without personal participation, then ensuring compliance with the enterprise architecture becomes an issue. On the other hand, if you cannot provide projects with documented rules they must comply with, then ensuring compliance is of no use – you either have the knowledge already, by having participated personally, or anything will do, as no one really knows what projects should comply with. Nevertheless, compliance is a quantity one needs to measure. It measures how many rules of the total are fulfilled. Thus, total compliance means simply that all rules are conformed to. Consequently, it is vital that all rules themselves are measurable in order to be able to measure the ratio. Models are in that sense truly problematic as they do not provide any measurable aspects. Clearly, extensive use of models in enterprise architecture documentation does not promote the ability to enforce the architecture.

In fact, the exclusive use of architectural principles makes the proposed documentation framework very sound in this respect. Every rule is represented in such a form that is meant to be measurable in the first hand. Thereby we have answered the second part of our research question, and as a conclusion we can state that the proposed enterprise architecture documentation and representation framework indeed has those qualities that we quested for in the research question.

6.2 CONTRIBUTIONS, IMPLICATIONS AND APPLICATIONS

In writing this thesis, we have become convinced that the "center of gravity" in the field of enterprise architecture has been located. Instead of continuous business/technology alignment, it has been found – due to the typically less agile nature of IT when compared to business processes – that it is best for business to choose an operating model for IT. It provides IT enough stability to develop those capabilities focal to the chosen operating model. These capabilities, for their part, presume some properties and qualities, e.g. non-functional requirements, from the software architectures of existing and yet to be built information systems. It is these requirements that enterprise architecture documentation shall primarily convoy, no more or no less.

The most significant contribution of this thesis is of course the enterprise architecture documentation and representation framework. This novel framework is indeed designed from these bases. It has three rather distinctive additional characteristics. First, it utilizes synthesis of concerns instead of separation of concerns in places. Hence, users get a fuller picture reading any piece of

the documentation. The pieces, on the other hand, are bound pretty much the same way as software architects would typically divide a system complex into basic entities. That way there is a direct link between a real world entity and a piece of documentation, which in part facilitates adoption. Second, it uses diagrams in architectural templates not only to depict the logical structure of the solution, but also to name the architectural elements that are parts of the solution. Third, it uses a unique type of chart to connect business process tasks to information systems that are used in execution of the tasks. By augmenting the information with information system aspects, like client type or client's physical location, that are relevant for the sought-after IT capabilities, users are able to quickly estimate whether a particular change in business process is IT-wise feasible.

Although the documentation framework itself is autonomous and basically any enterprise architecture framework could be adapted to use it, in practice there would be rather large gaps and discrepancies with many. The Gartner framework is conceptually quite close and thereby moderate adaptation is probably adequate, but all other frameworks would need extensive adaptation. We think this is an indication that either the established frameworks need to reshape to accommodate the new way of thinking or a brand new breed of frameworks must arise.

6.3 LIMITATIONS AND WEAKNESSES

The consequence of componentizing documentation is that instead of a handful of "monoliths" there are a number of smaller pieces of documentation. Apprehending this finer-grained structure takes awhile and users might get frustrated trying to locate the information needed until that happens. The big question, therefore, is whether too many users give in before they apprehend it. Things that can be done to speed up the learning process include making documentation browsing easier, typically by hyper linking the pieces, and providing documentation roadmaps.

Besides users the authors must understand the structure. An additional challenge for them is that the working method to produce the documentation changes from a model based on solo writing to group work. In this sense the decision to package several architectural views into same piece of documentation can be seen as a weakness, although at the same time the benefits of doing that must be stressed. Anyhow, a strong process is required to nurture the documentation creation.

Our experience is limited to public organizations' and businesses' IT. Probably the same limitation therefore exists in the proposed documentation framework – it might not work that well for say military organizations, where interoperability and other standards shape the architecture, or in cases where single factors, like longevity or predictable, strict response times, basically dictate the architecture.

CHAPTER VII
CONCLUSIONS

> ***enterprise architecture*** *Proclamation of the IT capabilities crucial to organization's business strategy, the derived principles governing the properties and qualities of software architectures fulfilling those capabilities, and the plan for their systematic development.*
>
> *(The author)*

In this thesis, we have approached the conventions and challenges related to the documentation and representation of enterprise architecture. As a conclusion, the contributions made are first summarized followed by a presentation of areas of future research.

7.1 SUMMARY OF CONTRIBUTIONS

In the previous chapter, we have already listed the major contributions made by this thesis by answering the research question. In this section we will summarize the contributions chapter by chapter.

7.1.1 INVESTIGATING THE RELATIONSHIP BETWEEN DOCUMENTATION AND USER COMPLIANCE

- Thus far the relationship between enterprise architecture documentation and user compliance has been largely neglected. Chapter 1 is a motivation for thoroughly investigating this relationship as it identifies that the business benefits of enterprise architecture initiatives will for the most part be lost unless the guidelines can be effectively and unambiguously communicated to all concerned parties.
- In chapter 2, a brief survey of existing work related to enterprise architecture documentation and representation is given. One notable detail is the categorization we introduced for enterprise architecture frameworks.
- Chapter 3 is an exposition of problems and obstacles, which were found during two real world enterprise architecture initiatives.

- Besides introducing the proposed documentation framework, chapter 4 lists and discusses the principles that we believe are deeply engaged with the relationship in question.
- The discussion continues more in-depth in chapter 5 as we evaluate our solution.

7.1.2 DEVELOPMENT OF A DOCUMENTATION FRAMEWORK

- Besides describing the proposed documentation framework, chapter 4 reveals the objectives as well as the principles used when developing it. Much larger contribution, however, is the framework – its structure and elements in particular. Many of the documentation conventions it includes are novel.
- Chapter 5 contains some preliminary results from using the framework.

7.2 FUTURE WORK

All research raises our awareness of issues that we do not truly understand, yet. This thesis is no exception. Below, we will briefly discuss some of the areas that require further research:

- We have claimed that written documentation is the best method to convoy enterprise architecture guidelines, but we currently do not have any evidence to support that. Empirical validation of written documentations effectiveness in contrast to, for instance, model-orientated or application template approaches is needed.
- We also claimed that an extremely cut down documentation style is more efficient and more unambiguous than a more descriptive one. This claim ought to be confirmed as well, perhaps using existing studies and research methods from the discipline of communication psychology.
- Another area that requires empirical validation is how well the structure of the framework and the form of documentation produced support the needs of various types of organizations and other concerned parties. This is a huge, but exceptionally significant, area due to the great variety of organizations of different size and structure and way of organizing their IT.
- New styles of software architectures, like service-oriented architecture, typically have a rather different pattern of emphasis on architectural issues than conventional architectures. The flexibility that calls for from the framework as well as the conceptual structures that support it offers still another area of challenge.
- As the framework produces documentation of repeated form and structure, automating the document generation could improve productivity by, for instance, making it easier to update material or enabling documentation to be output in many formats.

REFERENCES

Allega, P.J. (2005) *Architecture Framework Debates Are Irrelevant*. Research G00127331, Gartner.

Bredemeyer (2001) *Action Guides for the Enterprise Architect*. Bredemeyer Consulting.

CIO Council (1999) *The Federal Enterprise Architecture Framework Version 1.1*. Chief Information Officers Council.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., and Stafford, J. (2003) *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, Boston, MA.

Department of the Treasury CIO Council (2000) *Treasury Enterprise Architecture Framework Version 1*. Treasury CIO Council, Department of the Treasury.

D´Souza, D.F., and Wills, C. (1998) *Objects, Components, and Frameworks with UML : The Catalysis Approach*. Addison-Wesley, Reading, MA.

Fowler, M. (1997) *Analysis patterns: reusable object models*. Addison-Wesley, Reading, MA.

Fowler, M. (2003) *Patterns of Enterprise Application Architecture*. Addison-Wesley, Boston, MA.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994) *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, Reading, MA.

Hirvonen, A. (2005) *Enterprise Architecture Planning in Practice: the Perspectives of Information and Communication Technology Service Provider and End-user*, Ph.D. Thesis, Department of Computer Science and Information Systems, University of Jyväskylä.

Hirvonen, A., and Pulkkinen, M. (2004) A Practical Approach to EA Planning and Development: the EA Management Grid. *Proceedings of the 7th International Conference on Business Information Systems*, pp. 284-302.

Hofmeister, C., Nord, R., and Soni, D. (2000) *Applied Software Architecture*. Addison-Wesley, Reading, MA.

IEEE (2000) *Recommended Practice for Architectural Description of Software-Intensive Systems*. IEEE Std 1471-2000, IEEE.

Kruchten, P. (1995) Architectural Blueprints – the 4+1 View Model of Software Architecture. *IEEE Software*, 12 (6), pp. 42-50.

Kähkipuro (2005) Arkkitehtuurit – pelastusko tietotekniikkahaasteisiin. *Systeemityö*, 12(3), pp. 5-7.

Lindström, Å. (2006) On the Syntax and Semantics of Architectural Principles. *Proceedings of the 39th Hawaii International Conference on System Sciences 2006*, IEEE Computer Society.

Macehiter, N., and Ward-Dutton, N. (2005) *On IT-business alignment.* Macehiter Ward-Dutton.

Mack, R., and Frey, N. (2002) *Six Building Blocks for Creating Real IT Strategies*. Strategic Analysis Report R-17-3607, Gartner Research.

Malan, R., and Bredemeyer, D. (2001) *Defining Non-Functional Requirements*. White Paper, Bredemeyer Consulting.

Malan, R., and Bredemeyer, D. (2002) Less is More with Minimalist Architecture. IT Pro, September/October 2002, pp. 46-48, IEEE.

Malan, R., and Bredemeyer, D. (2005) *Enterprise Architecture as Strategic Differentiator*. Enterprise Architecture Executive Report, 8(6), Cutter Consortium.

Mannion, M., and Keepence, B. (1995) SMART Requirements. *ACM SIGSOFT Software Engineering Notes*, 20(2), pp. 42-47.

Naugher, L., and Rosser, B. (2002) *Talking Architecture: A Framework of Frameworks*. U.S. Symposium/ITxpo, Gartner.

Pulkkinen, M. (2006) Systemic Management of Architectural Decisions in Enterprise Architecture Planning. Four Dimensions and Three Abstraction Levels. *Proceedings of the 39th Hawaii International Conference on System Sciences 2006*, IEEE Computer Society.

Ranganathan, P., and Jouppi, N. (2005) Enterprise IT Trends and Implications for Architecture Research. *Proceedings of the 11th Int'l Symposium on High-Performance Computer Architecture*, IEEE Computer Society.

Robertson, B., and Sribar, V. (2002) *The Adaptive Enterprise: IT Infrastructure Strategies to Manage Change and Enable Growth*. Addison-Wesley / Intel Press, Hillsboro, OR.

Ross, J.W. (2003) *Creating a Strategic IT Architecture Competency: Learning in Stages*. MIT Sloan CISR Working Paper No. 335, Massachusetts Institute of Technology.

Ross, J.W., and Westerman, G. (2003) *Architecting New Outsourcing Solutions: The Promise of Utility Computing*. MIT Sloan CISR Working Paper No. 337, Massachusetts Institute of Technology.

Ross, J.W. (2005) Forget Strategy: Focus IT on your Operating Model. *MIT Sloan CISR Research Briefing 2005*, Vol. V, No. 3C.

Rozanski, N., and Woods, E. (2005) *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, Reading, MA.

Schekkerman, J. (2004) *How to survive in the jungle of Enterprise Architecture Frameworks*. Trafford Publishing, Victoria, BC.

Schekkerman, J. (2005) *Trends in Enterprise Architecture 2005: How Are Organizations Progressing?* Institute For Enterprise Architecture Developments.

Schulman, J. (2004) *Patterns Bridge Business and Technology Architectures*. Research G00123459, Gartner.

Shaw, M., and Garlan, D. (1996) *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, Englewood Cliffs, NJ.

Simsion, G. (2005) *What's Wrong With the Zachman Framework?* The Data Administration Newsletter (TDAN.com).

Sowa, J.F., and Zachman, J.A. (1992) Extending and Formalizing the Framework for Information Systems Architecture. *IBM Systems Journal*, 31 (3), pp. 590-616.

Tang, A., Han, J, and Chen, P. (2004) *A Comparative Analysis of Architecture Frameworks*. Technical Report SUTIT-TR2004.01, Swinburne University of Technology.

The Open Group (2003) *TOGAF (The Open Group Architecture Framework) Version 8.1 "Enterprise Edition"*. The Open Group.

Zachman, J.A. (1987) A framework for information systems architecture. *IBM Systems Journal*, 26 (3), pp. 276-292.