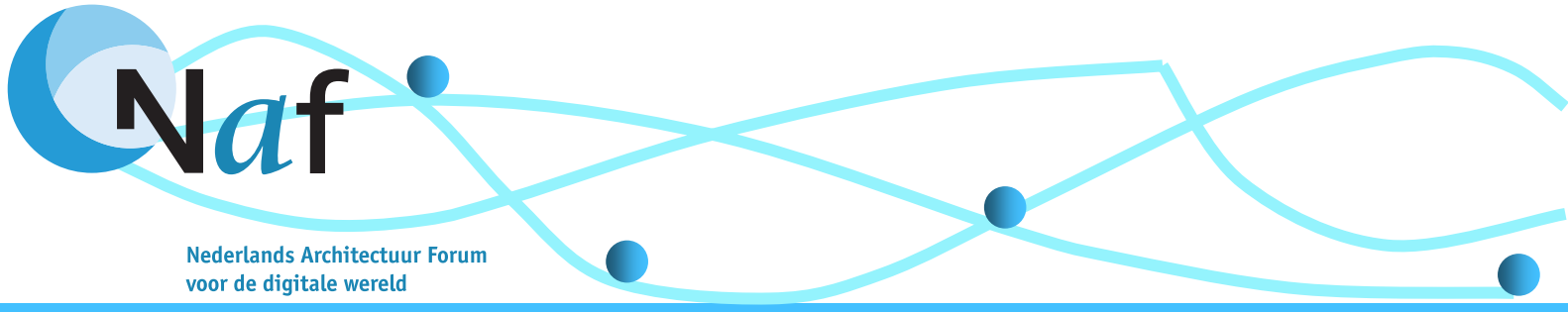


Competences of IT Architects

Roel Wieringa
Pascal van Eck
Claudia Steghuis
Erik Proper

NAF – Netherlands Architecture Forum for the Digital World

The field of architecture in the digital world uses a plethora of terms to refer to different kinds of architects, and recognises a confusing variety of competences that these architects are required to have. Different service providers use different terms for similar architects and even if they use the same term, they may mean something different. This makes it hard for customers to know what competences an architect can be expected to have.

This book combines competence profiles of the NGI Platform for IT Professionals, The Open Group Architecture Framework (TOGAF), as well as a number of Dutch IT service providers in a comprehensive framework. Using this framework, the book shows that notwithstanding a large variety in terminology, there is convergence towards a common set of competence profiles. In other words, when looking beyond terminological differences by using the framework, one sees that organizations recognize similar types of architects, and that similar architects in different organisations have similar competence profiles. The framework presented in this book thus provides an instrument to position architecture services as offered by IT service providers and as used by their customers.

The framework and the competence profiles presented in this book are the main results of the special interest group "Professionalisation" of the Netherlands Architecture Forum for the Digital World (NAF). Members of this group, as well as students of the universities of Twente and Nijmegen have contributed to the research on which this book is based.

About the authors

Prof.dr. Roel Wieringa and dr. Pascal van Eck are Full Professor and Assistant Professor in the Information Systems Group of University of Twente, The Netherlands. Claudia Steghuis M.Sc. is Senior Consultant enterprise architecture at Capgemini. Prof.dr. Erik Proper is Principal Consultant enterprise architecture at Capgemini and Full Professor of Information Systems at Radboud University Nijmegen, The Netherlands.

Roel Wieringa
Pascal van Eck
Claudia Steghuis
Erik Proper

Competences of IT Architects

2nd edition, 2009

Nederlands Architectuur Forum
voor de digitale wereld

Roel Wieringa
Department of Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands
Email: r.j.wieringa@utwente.nl

Pascal van Eck
Department of Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands
Email: p.vaneck@utwente.nl

Claudia Steghuis
Capgemini Nederland BV
Papendorpseweg 100
3528 BJ Utrecht
The Netherlands
Email : claudia.steghuis@capgemini.com

Erik Proper
Capgemini Nederland BV
Papendorpseweg 100
3528 BJ Utrecht
The Netherlands
and
Radboud University Nijmegen
Toernooiveld 1
6500 GL Nijmegen
The Netherlands
Email: e.proper@acm.org

Second edition (2009) published October 2009 by NAF – Nederlands Architectuur Forum voor de digitale wereld, www.naf.nl.

First edition (2008) published by Academic Service, ISBN 978-90-12-58087-8.

Typeset in Georgia and Verdana for on-screen reading and printing on A4 and letter paper.

Cover design and book layout: Pascal van Eck

ISBN: 978-94-90568-01-6.
NUR: 982.

Table of Contents

1 Introduction 1
2 Current IT Architect Profiles 3
2.1 A preliminary classification of architecture disciplines 3
2.2 Competences according to NGI 4
2.2.1 Personal competences 6
2.2.2 Professional competences 8
2.3 Competences according to The Open Group 10
2.4 Competences according to six large Dutch employers 13
2.4.1 Roles model 13
2.4.2 Tasks model 14
2.4.3 Competences model 16
3 A Framework for IT Architecture 19
3.1 Systems and emergent properties 19
3.2 System architecture 20
3.3 Architecture views 21
3.3.1 System aspects 21
3.3.2 System composition 22
3.3.3 Composition in three worlds 23
3.3.4 Layering 24
3.3.5 System phases 26
3.3.6 Description refinement 26
3.4 The GRAAL framework 29
3.5 Comparison with other frameworks 29
3.5.1 Zachman 29
3.5.2 The four-domain architecture 30
3.5.3 ArchiMate 31
3.5.4 The 4+1 model 32
3.5.5 Henderson and Venkatraman, Maes, and IAF 32
3.5.6 TOGAF 33
4 Architecture Disciplines 35
4.1 Basic architecture disciplines 35
4.2 Frequently occurring disciplines in practice 36
5 A framework for competences 39
5.1 Competences and proficiency levels 39
5.2 The competence pyramid 40
6 IT architect competences 43
6.1 Professional competences 43
6.2 General professional competences 47
6.3 Cultural characteristics 48

6.4 Personality characteristics 48
7 Summary and recommendations 51
8 Bibliography 53
9 Appendix 55

Preface

In this book we report on research conducted in the past few years under the auspices of the NAF[1] into competences of IT architects. It is not possible to give a name to the profession of IT architect without raising protests from more than a few of the professionals who, in this book, we refer to as "IT architects". Some insist on the label "business-IT architect", but others think "enterprise architect" is a much better term. Some even use the term "digital architect" to stress the fact that these architects supposedly shape the "digital world". Many are more liberal in their acceptance of labels for the profession, but "IT architect" just happens to be the one term that they definitively disagree with. Even more, some people in the community would argue that there is nothing wrong in using "old-fashioned" term "information architect". Quarrels about names tend to be as emotional as they are pointless. In this book we aim at making the relevant distinctions visible, but we do not propose a definitive set of labels from the different disciplines within the architect profession. We will use a consistent set of terms, but do not pretend to tell others to use the same set of terms. We, more or less arbitrarily, use "IT architect" as the most general label for the profession, within which all the others fall.

Having said this, the intended audience of the book consist of IT architects who reflect on their profession, and of those who hire and/or manage IT architects. For these people it is important to have a clear picture of the competences required by IT architects. Different companies all use their differently defined IT architect job roles and architecture disciplines, and there is no uniformity of terminology or competence profiles,

We have attempted to bring some order in this multitude of views by analyzing competence profiles of different companies, analyzing published guidelines of professional bodies, by interviewing architects and their managers and by conducting a survey of 139 IT architects assembled at the national architecture conference[2] in 2004. We analyzed the results in terms of an architecture framework and of a classification of professional competences and of personality characteristics.

We hope the result will be useful to companies who intend to hire or appoint employees or consultants in the role of IT architect. Because we compare and explain a number of different profiles used in different companies, our results should bring some order in the sometimes confusing terminology.

Different parts of the research for this book were done by Henk Blanken, Pascal van Eck, Corrie Huijs, Erik Proper, Claudia Steghuis, Koen Voermans, and Roel Wieringa. Henk Blanken, Pascal van Eck and Erik Proper explored competence profiles of architects

[1] Netherlands Architecture Forum, www.naf.nl

[2] Landelijk Architrectuurcongres, LAC 2004.

in a brown paper session with about 15 architects of NAF member organisations. Pascal van Eck and Corrie Huijs analyzed competence profiles of some companies and of the NGI[3]. Claudia Steghuis and Koen Voermans interviewed IT architects and conducted a survey at LAC 2004. Roel Wieringa collected the results and analyzed them. The authors are grateful to Frank Baldinger, Chair of the Netherlands Architecture Forum (NAF), for his persistent and patient reminders to finish the preparation of this book.

About the authors:

Roel Wieringa

Roel Wieringa is full professor of information systems at the University of Twente, The Netherlands, and is chair of the Information Systems group. The Information Systems group performs research in business-IT alignment, IT-enabled value networks, and mobile and context-aware service provision and information security in business networks. His recent publications concern the mutual alignment of IT services and IT architecture in value webs and the role of business problems in structuring IT requirements specifications. Wieringa shares a joint responsibility for the Masters of Science in Business Information Technology, which teaches the principles and practice of business-IT alignment, process management, architecture and governance in business networks.

Pascal van Eck

Pascal van Eck received his diploma in Computer Science (MSc degree) from the Free University of Amsterdam in April, 1995. From May, 1995 until January 2000, he worked as a research assistant at the Artificial Intelligence Department of the Free University on a PhD thesis on a formal, compositional semantic structure for the dynamics of multi-agent systems. Starting February, 2000, he works at the Information Systems Group of the Faculty of Computer Science, University of Twente, as an assistant professor. His research interests include alignment of business and ICT architectures at the enterprise level, with a focus on case studies in the Dutch financial services industry and large-scale transaction processing, as well as cross-organisational integration of enterprise applications.

Claudia Steghuis

Claudia Steghuis is senior consultant at Capgemini with about two years of experience in enterprise architecture. She is a TOGAF certified architect. Claudia obtained a master degree in Business information technology, discipline BIT Architecture at the University of Twente. Her MSc research focussed on service granularity within service oriented architecture. She is co-author of the book Enterprise Architecture – Creating Value by Informed Governance.

[3] Nederlands Genootschap van Informatici.

Erik Proper

Erik Proper is Principal Consultant at Capgemini and Professor in Information Systems at the Radboud University Nijmegen. At the Radboud University he leads the Tools for Enterprise Engineering research group. Erik has a mixed industrial and academic background. His interests lie mainly in the field of conceptual modelling, enterprise modelling, enterprise engineering and enterprise architecting. He was co-initiator of the ArchiMate project, and currently also serves on the board of the ArchiMate foundation as well as the Netherlands Architecture Forum (NAF).

1 Introduction

There is a plethora of titles to refer to IT architects, and a confusing variety of competences that these architects are required to have. Different consultancy companies use different terms for similar architect roles, and even if they use the same term they may mean something different. This makes it hard for clients to know what competences an IT architect offered by a consultancy company can be expected to have. There have been some attempts at standardization, such as that of the NGI in the Netherlands for the entire field of informatics, and of The Open Group for the field of IT architecture. So far, these attempts have added more variety to the set of IT architect titles but the community has not yet settled on one accepted set of titles and competences.

It is time to take stock of where we are in the field and to analyze if perhaps there is more agreement than what meets the eye if one reviews the titles used by different companies. In this monograph we will analyze some of the underlying structures in the seemingly confusing variety of IT architect profiles

In Chapter 2 we take stock of the current variety of IT architect profiles. It turns out that any profile presupposes an architecture framework, which is a set of dimensions along which architectures can be described, and which therefore also classifies knowledge and skills of the people who design these architectures. In Chapter 3 we present a unified framework that represents the underlying structure of frameworks that we found in different companies, and of well-known frameworks such as that of Zachman (1987), Sowa and Zachman (1992) and of the Open Group (TOGAF).This is our first step towards uncovering the underlying structure of the variety of IT architect profiles. Using the framework of Chapter 3, it is a simple matter in Chapter 4 to identify a number of architecture disciplines and to show how this corresponds to a variety of disciplines found in different organisations and standardization efforts.

Each IT architecture disciplines requires its own competences, and to classify these competences we describe in Chapter 5 a framework that classifies competences into technical, professional, cultural and personal competences. We then use this in Chapter 6 to classify competences of IT architects of each of the disciplines that we listed earlier. This yields a set of profiles that, if we have done our work well enough, should look familiar to many people, and that they can relate to the set that they work with in their own company. Our competence profiles should therefore be useful as bridge between the profile sets of different companies.

We should point out explicitly that we have *not* shown that architects with these competences will design excellent architectures. Architecture design is a social process of which the outcome does not depend on the effort of a single person. Neither do we claim that, even when an architect had performed their work alone, the outcome will be a good architecture. We have no empirical evidence of the existence of any link between the competences of IT architects and the quality of the architectures that they design. This is not to say that there is no such link; we think there is such a link, but we have done no empirical research to indicate its existence. None of the sources that we reviewed provides this evidence either. In order to produce evidence of such a link, we should select archi-

tects with the competences that we describe, have them design architectures, and then evaluate the architectures; if they are good, we should then show that the quality of these architectures derives from the competences of the architects rather than some other factor, such as the quality of documentation or the nature of the example domain. We are not aware of any research of this kind, and it was not our intention to perform this research.

What we have done instead is to review *claims* made by companies, standardization bodies and IT architects about the competences required for IT architecting. In this monograph we analyze and systematize these claims. Our own claim is that we have given a fair systematization of the material, and we motivate this by making our analysis explicit. This does mean that we have uncovered a consensus in the field, and to the extent that this consensus is based on experience, this does tell us something about what is required to be a good architect.

2 Current IT Architect Profiles

This chapter presents currently existing competence profiles of IT architects according to several sources. As can be expected, each of these sources describes different competence profiles for different architecture disciplines. We therefore introduce, in Section 2.1, a preliminary classification of architecture disciplines. This classification is in advance of the classification presented in Chapter 4. In Section 2.2, we start with the first source by presenting an analysis of about five different architecture-related professions distinguished by the Dutch Society of Information Professionals (NGI). Section 2.3 presents similar profiles that have been defined by The Open Group as part of The Open Group Architecture Framework (TOGAF). Section 2.4 presents competence profiles that have been found in interviews with six large Dutch IT employers.

2.1 A preliminary classification of architecture disciplines

As competence profiles differ for different architecture disciplines, we need a classification of architecture disciplines to be able to present our findings. We use the classification of Steghuis, Voermans and Wieringa (2005), as their report forms the basis of Section 2.4. This classification is introduced in another report (Voermans, Steghuis and Wieringa 2005) and is derived from the GRAAL framework that is presented in Chapter 3 of this book.

Steghuis et al. (2005) differentiate between architecture disciplines by considering the types of elements that constitute an enterprise architecture. These elements are present in an organisation because they are useful: every element provides useful functions, or, in other words, provides services, to another element. The service provider – service consumer relationship between elements creates a hierarchy. Steghuis et al. distinguish five layers in this architecture:

- Business environment: entities in the environment of the organisation to which the organisation delivers products and/or services. For commercial companies, the most important type of elements of the business environment are their customers.
- Business: the products and services that the organisation produces for its environment, the processes that create these products and services, the employees who perform those processes, the formal and informal relations between those employees, etc.
- Enterprise software systems (called 'business systems' by Steghuis et al.): organisation-specific software systems that support the processes and people in the business.
- Software infrastructure: software systems that are not specific for the organisation, such as operating systems, database management systems, email servers, etc.
- Physical infrastructure: processors, disks, network routers, switches and cables, and all other physical objects that are needed to run the software systems that constitute the business systems and software infrastructure layers.

We can use this hierarchy to describe architecture disciplines: for instance, we can define an infrastructure architect as an architect whose main responsibility is creating and maintaining the architecture of the software infrastructure.

2.2 Competences according to NGI

Since the early 1980s, the Dutch Platform for ICT Professionals NGI maintains a collection of computing-related job descriptions. The current version dates from 2001 (Op de Coul 2001); we refer to this version as 'the NGI report' in this book. NGI provides an inventory of possible roles of informatics professionals, the functions of these roles in an organisation, the tasks performed by these roles in these functions, and the competences required by those tasks. One task may be part of various functions, and may require several competences; and one competence may be required by several tasks. The NGI recognizes five types of jobs with the term 'architecture' in their name. In 2004, Koen Voermans and Pascal van Eck analysed the competences of the architects recognized by the NGI and compared these with the competences of a number of designer professions on the one hand and of the CIO profession on the other, all as described by the NGI.

The five types of architects are[4]:

- **Information Architect.** "An Information Architect formulates – usually starting from architecture principles – the information architecture, with a strong emphasis on optimalisation of business processes. Special attention is given to integration of information systems within the organisation, as well as with system of customers, etc."
- **Data Architect.** "The Data Architect analyses business processes, determines which data and information is needed for optimalisation of these processes, and translates these into a data architecture, which is part of the information architecture." (According to the NGI report, an alternative name for this function is data manager.)
- **Software architect.** "The Software Architect develops - often starting from architecture principles – the information systems architecture. The Software Architect is usually involved in the development of complex information systems (think of ERP-solutions) to obtain a thorough and optimal structure of an application. Is also usually the ultimate responsible for the design and quality of a system."
- **Technical Infrastructure Architect.** "Usually starting from architecture principles, the Technical Infrastructure Architect designs, based on the information architecture and the information- and automation plans, the structure of the technical infrastructure (the structure of networks, the specification of the components that are part of it such as network devices and servers, including issues such as network protocols and systems software). Usually, the Technical Infrastructure Architect is also involved in selecting those network devices and systems software". (Alternative name according to the NGI report: Advisor Technical Infrastructure.)
- **Network Architect.** "The Network Architect develops, based on the information architecture and the information and automation plans, the structure of a network, LAN or WAN (including the specification of the components (network devices) and protocols of the network). Usually, the Network Architect is also involved in the selection of

[4] Quoted text are English translations of the descriptions in (Op de Coul, 2001).

those (network) control, management and systems software. The Network Architect is a specialization of the Infrastructure Architect."

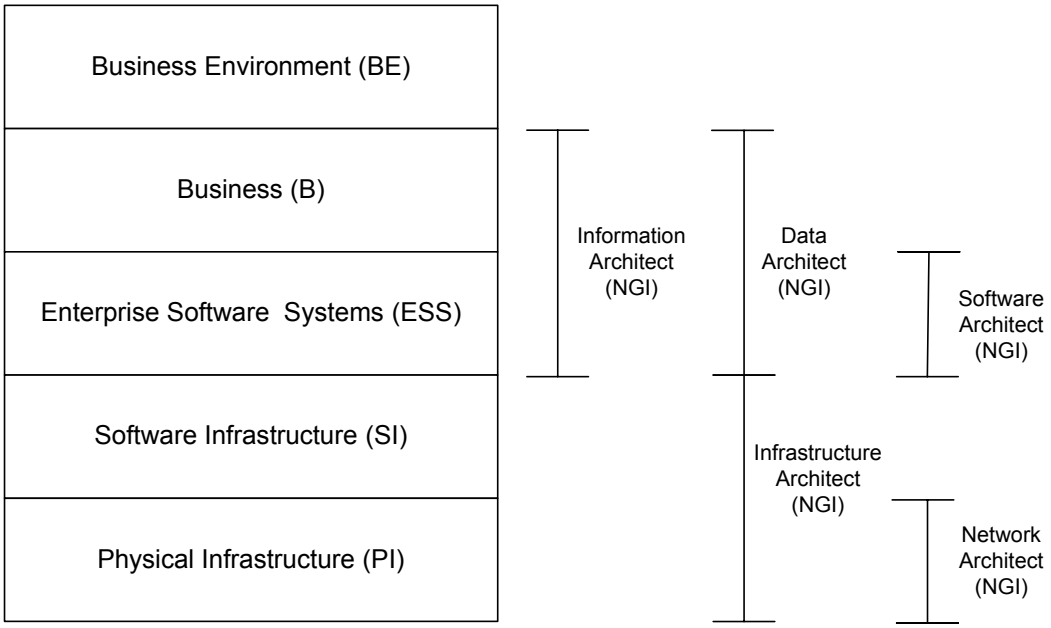


Figure 1 Classification of the architecture disciplines described in the NGI report

In Figure 1, we compare these architecture disciplines according to the classification presented in Section 2.1.

In the NGI report, a function in an organisation is nothing more than a set of tasks. Thus, for each of the five architecture disciplines mentioned before, there is a list of tasks. We present these lists (in Dutch) for four of the five disciplines in Table 11 in the appendix (the fifth function, the Network Architect, is according to the NGI report, a specialisation of the Technical Infrastructure Architect; in the rest of this chapter, we do not mention this discipline at all). In the NGI report, for all 71 functions together, there are 142 different tasks, of which 41 are present in Table 11 in the appendix.

Next, in the NGI report, for each task, a set of professional competences and a set of personal competences are defined. For example, for the Information Architect function (which, according to Table 11 in the appendix, has 13 tasks), there are 13 lists of professional competences and 13 lists of personal competences. The 13 lists of professional competences are overlapping: a number of competences are mentioned in more than one list. The same holds for the 13 lists of personal competences. The 13 lists of professional competences together list 108 competences (which amounts to an average of 8-9 competences per task); of which 20 are unique (thus, each competence appears on average in 5-6 tasks). For all 142 different tasks in the NGI report, there are 41 different professional competences, which are categorized in 5 main categories. Table 12 in the appendix lists all 20 professional competences that appear in the description of the Information Architect function. We visualize the competence profiles thus provided by the NGI report in the following two sections, starting with personal competences as their visualization is slightly less complex.

2.2.1 Personal competences

Figure 3 shows a visualization of the personal competences of a number of architecture disciplines and related disciplines according to the NGI report. As stated above, in the NGI report, each role has a number of functions, and each function a number of tasks, for which competences are required. If the same competence is required for different tasks of the same function, and the same task is performed by different functions of the same role, then the same competence can be mentioned several times for one role. This allows us to count the number of times a competence is mentioned for a role. We use this in two ways in the visualization.

First, the diagram consists of a number of rectangles of different sizes, organized in columns that cluster related competences. The size of each rectangle indicates the 'importance' of the competence. This importance was measured by first counting the number of times a competence was mentioned per function, and then averaging this number over different functions. So the diagram says that the analytical competence was, on the average, mentioned most often for different functions.

Second, each rectangle contains a number of "stick men" of different colours (and with different letters to facilitate black-and-white printing), where each stick man is labelled by a number. The colour/letter of a stick man refers to a role, as indicated by the legend. The number of each stick man indicates how frequently this competence was mentioned for this function. So the diagram shows that according to the NGI, a data architect (blue/letter D) must be first of all analytical (in fact, most architects must have this competence). Second in importance for data architects is abstraction, then comes a methodical way of working, accuracy and creativity, in that order. According to the NGI, a CIO must first of all have business awareness, be able to judge situations, be analytical, be communicative, and be able to abstract—in that order.

Table 1 compares the top 5 competences listed in Figure 3.

NGI Technical infrastructure architect	**NGI Data architect**	**NGI Information architect**	**NGI Software architect**
Analytical	Analytical	Analytical	Analytical
Accurate	Abstraction	Business awareness	Methodical
Business awareness	Methodical	Methodical	Accuracy
Methodical	Accuracy	Judgment	Creativity
Judgment	Creativity	Creativity	Communicative

Table 1 NGI architecture disciplines: personal competences compared

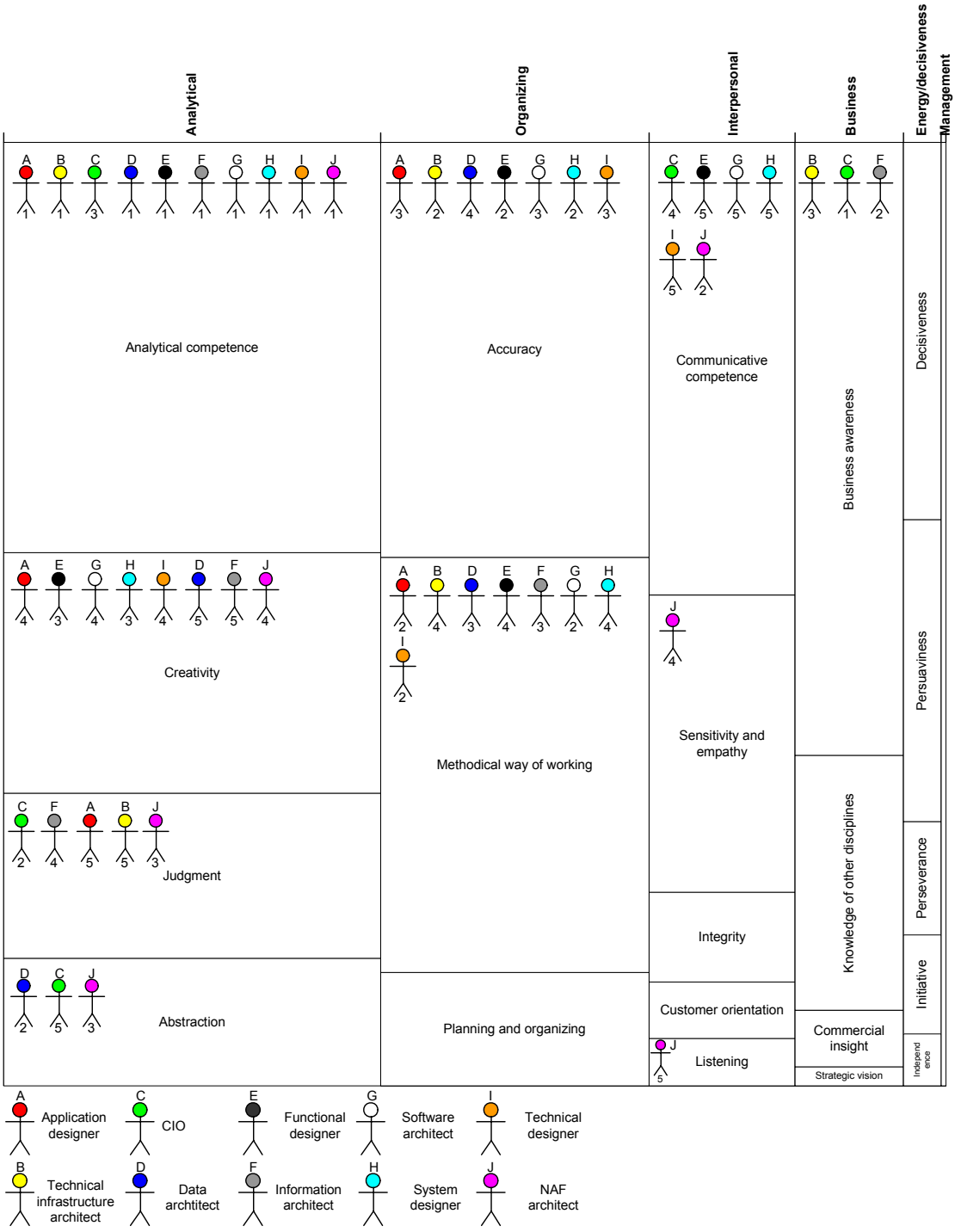


Figure 2 Visualization of professional competences in the NGI report

There is consensus that all architects must be analytical. They must also all be methodical, in various degrees of importance, and be creative, slightly less important. Creativity is not among the top 5 competences of infrastructure architects. Surprisingly, infrastructure architects must be business aware just as information architects must be. Accuracy is an important competence for all NGI architects except information architects. For NAF ar-

chitects, communicative skills are regarded as very important. This is not important for NGI architects except, surprisingly, for software architects.

2.2.2 Professional competences

Turning to professional competences according to the NGI report (Figure 3) we divided these into an information systems column and an organisation column, with a narrow documentation column added. The rectangles have also been organized into layers that roughly correspond to knowledge domains, such as methods and techniques, infrastructure and IT management. Analyzing these domains, we see that they consist of technology domains (servers, networks, database management systems) and various roles/functions for these domains: designing, building, managing. Maintenance is considered at the technical as well as organisational level (technical change and organisational change) and there are some general competences related to methods and techniques for analyzing and designing IT and organisations. The business competences are clustered in some knowledge areas such as administrative organisation, management science and organisation science.

Comparison of NGI architects yields the following table of top 5 competences.

NGI Infrastructure architect	**NGI Data architect**	**NGI Information architect**	**NGI Software architect**
Possibilities of IT	Possibilities of IT	Management science	Possibilities of IT
IT analysis methods	Administrative organisation	Possibilities of IT	IT analysis techniques
Server management	Management science	Administrative organisation	IT design techniques
Network management	Organisational science	Organisational science	Administrative organisation
IT design methods	Organisation analysis methods	Organisation analysis techniques	Management science
Server technology	IT analysis methods	Organisation design techniques	Organisation science
Network technology	Quality management	Quality management	Management of information systems

Table 2 Professional competences of NGI architects

All architects must be able to see possibilities of IT for stakeholders. NGI infrastructure architects must be knowledgeable of infrastructure technology (construed as server- and network technology). The other NGI architects must have considerable organisational competences, such as administrative organisation, management science and organisation science. Surprisingly, this is also true of NGI software architects. IT analysis methods are important for all architects except NGI information architects, and IT design methods are important for infrastructure and software architects.

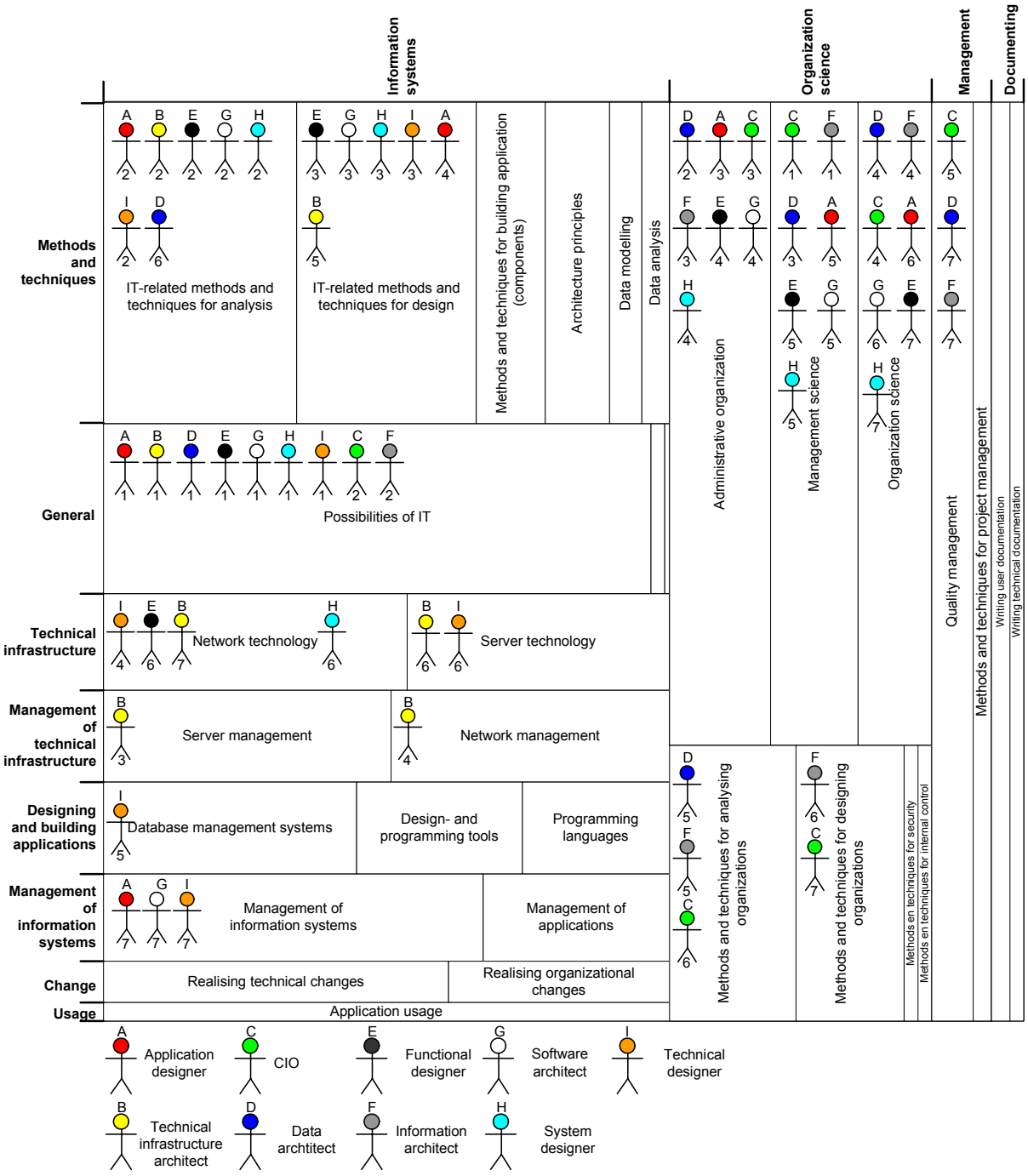


Figure 3 Visualization of professional competences in the NGI report

The technical architect competences emphasized by the NGI include the competence to see possibilities of IT for stakeholders, and, depending on the architect's role, competences in analyzing and designing IT and/or organisations. The NGI identifies some infrastructure domains for infrastructure architects (network, server) and some organisational domains for the other architects (administrative organisation, management science and organisation science).

2.3 Competences according to The Open Group

The Architecture Forum of The Open Group, a worldwide consortium of architecture practitioners, has developed TOGAF (The Open Group Architecture Framework). Part 4 of TOGAF, the resource base, presents the TOGAF Architecture Skills Framework, which "provides a set of roles, skill, and experience norms for staff undertaking enterprise architecture work" (TOGAF, Ch. 30).

TOGAF identifies nine different architecture roles: the architecture board member, the architecture sponsor, the IT architecture manager, architects in four different areas (technology, data, application, and business), the program or project manager, and the IT designer. These roles require proficiency in skills in seven areas:

- "Generic skills". This area lists 8 skills, such as leadership, different communication skills, and logical analysis.
- "Business skills and methods". This area contains 11 skills, such as strategic planning, business case development, and budget management.
- "Enterprise architecture skills". This area contains 17 skills, such as business modelling, architecture principles design, and application design.
- "Program or project management". This area lists 5 skills, namely program, project, change and value management, and `managing business change'.
- "IT knowledge skills". This area lists 17 skills, such as programming languages, storage management, COTS, and migration planning.
- "Technical IT skills". This area lists 13 skills, such as security, systems and network management, graphics and imaging, and data interchange.
- "Legal environment". This area lists 5 skills, namely four law domains (contract, data protection, procurement and commercial law), and fraud.

The seven areas together list 76 skills. For each skill, TOGAF defines for each role the level at which the skill is needed. TOGAF distinguishes four skill levels, which are characterised as follows (taken verbatim from the TOGAF documents):

- Level 1 (Background): "Not a required skill though should be able to define and manage skill if required."
- Level 2 (Awareness): "Understands the background, issues, and implications sufficiently to be able to understand how to proceed further and advise client accordingly."
- Level 3 (Knowledge): "Detailed knowledge of subject area and capable of providing professional advice and guidance. Ability to integrate capability into architecture design."
- Level 4 (Expert): "Extensive and substantial practical experience and applied knowledge in the subject."

Altogether, TOGAF defines the architecture roles in the form of 684 skill levels (76 skills times 9 roles), presented in a sequence of tables (one for each skill area) in the TOGAF documents. We analysed these tables and summarize them as follows. Table 3 lists, for each skill area and role, the number of skills that the role needs to master at the expert level according to TOGAF. For example, the `generic skills' area contains 8 skills, 38% (3 skills) of which the architecture board member needs to master at the expert level. The

four architecture disciplines (IT Architecture Technology, IT Architecture Data, IT Architecture Application and IT Architecture Business) each need to master 63% (5 skills) of the 8 generic skills at the expert level, but these are not necessary the same skills. Table 4 has the same structure, but in this table, skills at the knowledge level and expert levels (the two highest levels) are taken together.

Percentage of competencies at expert level	Number of Skills	Architecture Board Member	Architecture Sponsor	IT Architecture Manager	IT Architecture Technology	IT Architecture Data	IT Architecture Application	IT Architecture Business	Program or Project Manager	IT Designer
IT Architect Roles										
Generic Skills	8	38%	25%	100%	63%	63%	63%	63%	75%	0%
Business Skills & Methods	11	18%	36%	64%	36%	45%	45%	73%	27%	0%
Enterprise Architecture Skills	17	0%	0%	82%	53%	47%	82%	59%	12%	0%
Program or Project Management Skills	5	20%	20%	60%	0%	0%	0%	60%	60%	0%
IT General Know ledge Skills	17	0%	0%	24%	65%	59%	65%	6%	0%	0%
Technical IT Skills	13	0%	0%	0%	92%	38%	31%	0%	0%	0%
Legal Environment	5	0%	0%	20%	0%	0%	0%	0%	20%	0%
Total	76	8%	9%	49%	54%	43%	51%	36%	20%	0%

Table 3 TOGAF competences at the expert level

Percentage of competencies at knowledge or expert level		Architecture Board Member	Architecture Sponsor	IT Architecture Manager	IT Architecture Technology	IT Architecture Data	IT Architecture Application	IT Architecture Business	Program or Project Manager	IT Designer
IT Architect Roles										
Generic Skills	8	88%	88%	100%	100%	100%	100%	100%	100%	25%
Business Skills & Methods	11	82%	100%	100%	91%	91%	91%	91%	100%	18%
Enterprise Architecture Skills	17	6%	6%	100%	100%	100%	100%	100%	24%	29%
Program or Project Management Skills	5	60%	60%	100%	100%	100%	100%	100%	100%	0%
IT General Know ledge Skills	17	0%	0%	100%	94%	100%	82%	41%	6%	100%
Technical IT Skills	13	0%	0%	100%	100%	100%	100%	46%	0%	92%
Legal Environment	5	80%	60%	40%	40%	40%	40%	60%	80%	0%
Total	76	32%	33%	96%	93%	95%	91%	74%	43%	50%

Table 4 TOGAF competences at the knowledge level or higher

According to Table 4, TOGAF is quite demanding: The IT Architecture Manager and three of the four architecture disciplines need to master almost all skills (more than 80%) in all but one skills area at the knowledge level or higher. It is therefore difficult to differentiate between the architecture disciplines: only the IT Architecture Business role stands out as this role may be considerably less skilful in the areas 'IT General Knowledge Skills' (41%) and 'Technical IT Skills' (46%). Table 3 is less uniform than Table 4. The 'broadest' architects according to TOGAF are the technology and application architects, who are the only ones required to master (slightly) more than half of all skills at the expert level.

Table 5 repeats Table 3 (TOGAF competences at the expert level), focusing on the four architecture disciplines and four professional competence areas. Within this focus, cells

with a value higher than 50% are highlighted. In Figure 4, we map the four professional competence areas to the architecture disciplines introduced in Section 2.1. In this figure, we assume that an architecture discipline covers a layer if this architecture discipline is required to master more than 50% of the skills of the knowledge area(s) associated with this layer. Thus, the IT Architecture Data discipline covers only the Enterprise Software Systems layer, as this discipline only needs to master more than 50% of the skills in one area (the 'IT Knowledge Skills' area), and this area is associated with only one layer, the Enterprise Software Systems layer.

Percentage of competencies at expert level	Number of Skills	Architecture Board Member	Architecture Sponsor	IT Architecture Manager	IT Architecture Technology	IT Architecture Data	IT Architecture Application	IT Architecture Business	Program or Project Manager	IT Designer
IT Architect Roles										
Generic Skills	8	38%	25%	100%	63%	63%	63%	63%	75%	0%
Business Skills & Methods	11	18%	36%	64%	36%	45%	45%	73%	27%	0%
Enterprise Architecture Skills	17	0%	0%	82%	53%	47%	82%	59%	12%	0%
Program or Project Management Skills	5	20%	20%	60%	0%	0%	0%	60%	60%	0%
IT General Knowledge Skills	17	0%	0%	24%	65%	59%	65%	6%	0%	0%
Technical IT Skills	13	0%	0%	0%	92%	38%	31%	0%	0%	0%
Legal Environment	5	0%	0%	20%	0%	0%	0%	0%	20%	0%
Total	76	8%	9%	49%	54%	43%	51%	36%	20%	0%

Table 5 Classifying TOGAF architecture disciplines

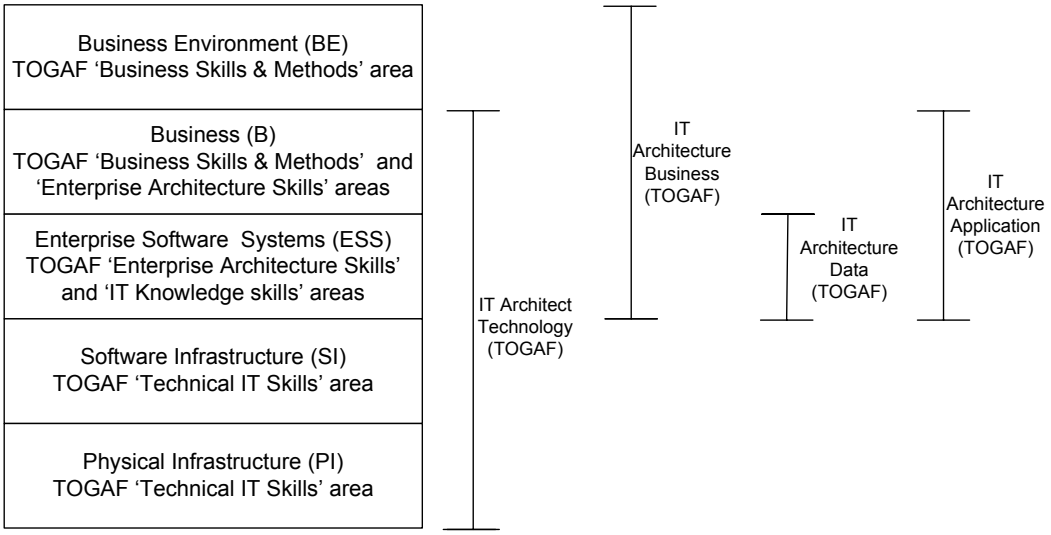


Figure 4 Classification of the TOGAF architecture disciplines

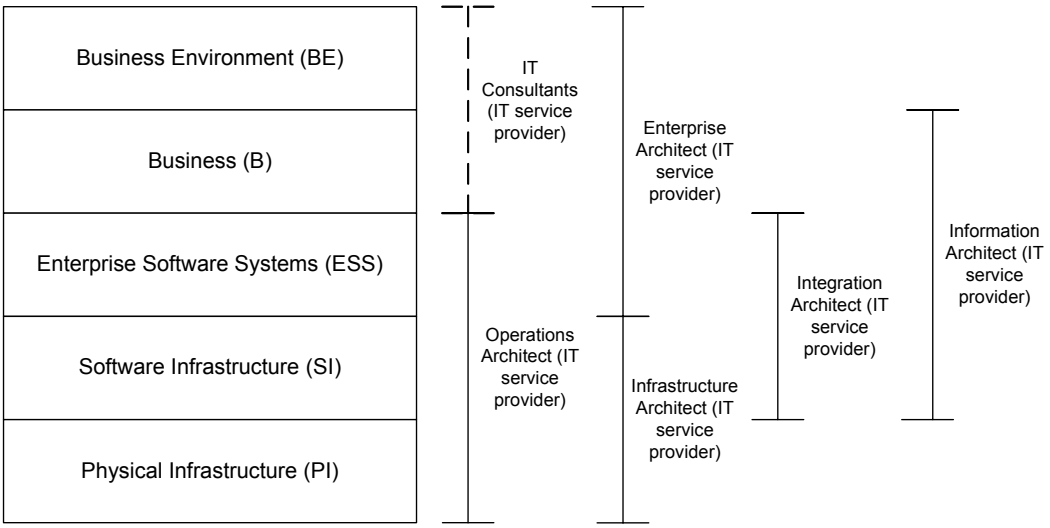
After presenting the skills tables, Chapter 30 of TOGAF discusses the generic roles and skills of the IT architect in terms of job descriptions. This part of the chapter does not introduce new skills or competences.

2.4 Competences according to six large Dutch employers

In 2005, Steghuis et al. (2005) interviewed six large Dutch employers of architects (a large multinational financial service provider, two consulting companies, two IT service providers and a multinational hardware and software vendor). For each of these six organisations, the interviewers determined the architecture roles identified in the organisation and the tasks and competences associated with these roles. They analyse their results by populating a 'roles model' and a 'tasks model' (both of which evolved in models used in later chapters in this book). We summarize this analysis in this section using the same structure: we present the populated roles model in Section 2.4.1, the tasks model in Section 2.4.2, and the competences model in Section 2.4.3. In Section 4.2, the results of this study are discussed further in the context of architecture disciplines observed in practice.

2.4.1 Roles model

Figure 5 maps the roles distinguished by three large organisations (an IT service provider, a consultancy company, and a bank/insurance company) to the typology introduced in Section 2.1. Steghuis et al. (2005) conclude that "As is clear from this picture, there is less agreement about first of all the names, the areas a role should cover, and the number of roles that are needed to cover all areas. Mostly because of these reasons it is very hard to extract a shared definition of roles from these data."



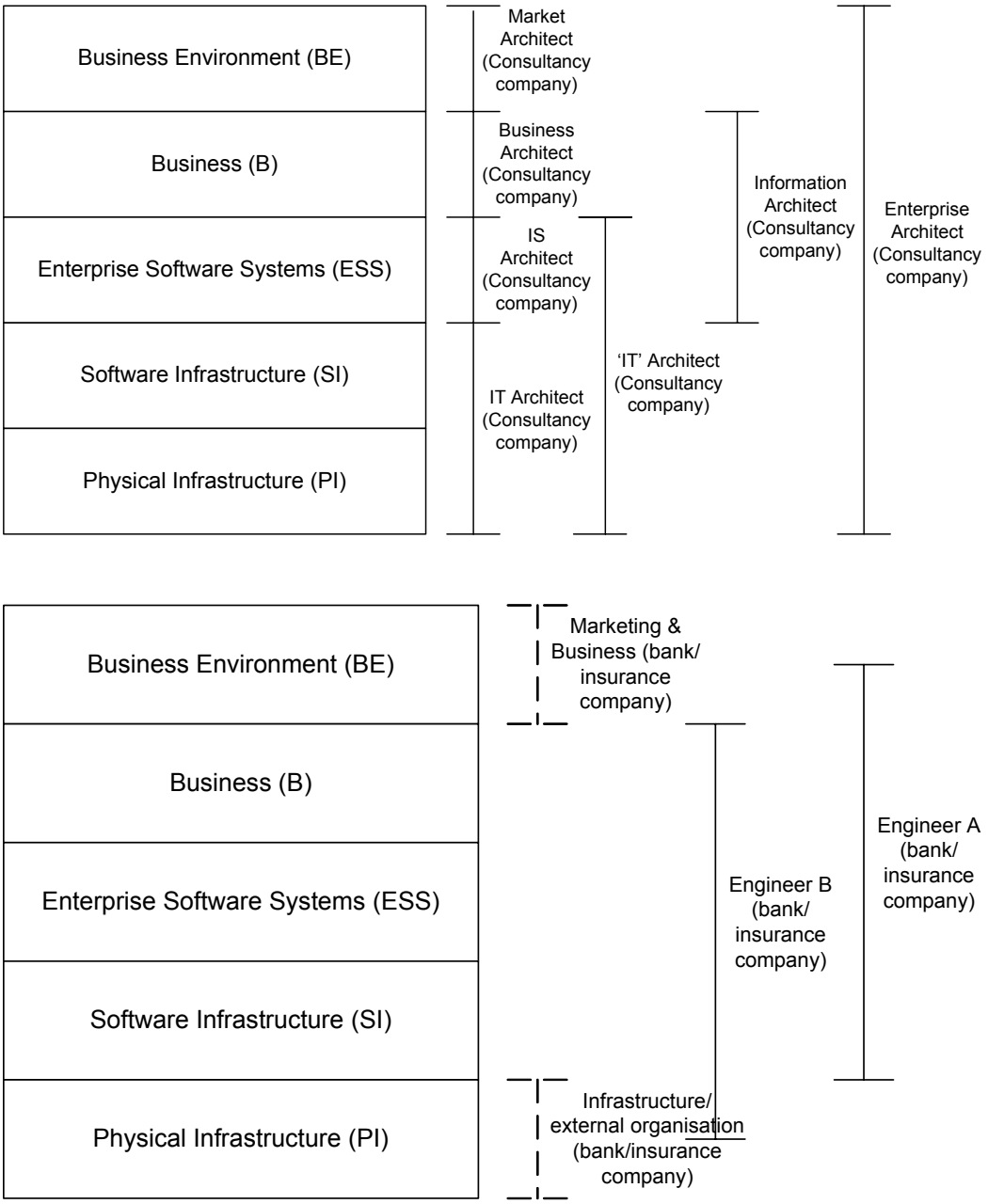


Figure 5 Architectural roles combined

2.4.2 Tasks model

Steghuis et al. (2005) further characterize the architecture disciplines found in the study by listing the tasks of each architecture discipline according to the interviews. This characterization is summarized in Table 6.

Role	Task
Engineer A (bank/insurance company)	Follows new developments of architecture both inside and outside the organisation. Identifies possible new opportunities for the organisation.
	Develops a vision and a strategy for the improvement of processes and the testing of processes, on a tactical and strategically level.
	Takes initiatives to improve technical strategies, working methods, processes, procedures and used methods
	Stimulates the use and uses modules for the design of information systems.
	Participates in the "off the shelf" product selection process. Intermediates between the customer and the IT organisation. Evaluates the product on compatibility with the ICT infrastructure and has in depth knowledge of "off the shelve" software
	Designs information and/or system architectures for complex domains. (Tactical and strategic level for engineer A)
	Designs and controls migration of large software products
	Does trend analysis on a tactical level
	Defines the scope and risk of projects based on the needs and wishes of the actors involved.
	Responsible for the realisation of a project in terms of money, time, customer satisfaction and quality.
Engineer B (bank/insurance company)	Analysis's needs and demands of new or existing products and formulates solution directions and alternatives. Primary focus is IT, but also organisation, Human Resources and processes are part of the analysis.
	Advises the project management and/or general management on the IT solutions, organisational procedures, people, and resources.
	Conducts a market research on possible "off the shelve" product solutions
	Checks ideas and designs on quality and organisational support
	Follows new developments of architecture both inside and outside the organisation. Identifies possible new opportunities for the organisation.
	Writes technical design documents, and specifications for complex software products. Designs and realises changes in complex automated information systems
	Stimulates the use and uses modules for the design of information systems.
	Participates in the "off the shelve" product selection process. Intermediates between the customer and the IT organisation. Evaluates the product on compatibility with the ICT infrastructure and has in depth knowledge of "off the shelve" software
	Designs information and/or system architectures for complex domains. (Tactical and strategic level for engineer A)
	Designs and controls migration of large software products
	Stimulates the continuity of information systems. Stimulates the pro-active removal of incidents, problems and weak points of systems
	Determines priorities for changes

	Periodically checks and performs emergency scenarios
	Defines the scope and risk of projects based on the needs and wishes of the actors involved.
	Responsible for the realisation of a project in terms of money, time, customer satisfaction and quality.
General (IT service provider 1)	Programme management
	Portfolio management
	Abstraction of problem statements
	Integration and execution
General (IT service provider 2)	Defines solutions to client business problems through the reasoned application of information technology
	Is responsible for the conceptual integrity of the solution
	Has extensive knowledge of systems, architectures, systems management, networking, network computing and application design techniques;
	Is able to identify, evaluate & select the elements of the solution which best meet the needs of the client organisation;
	Usually an architect has a 'T' shaped skill profile. This means that an architect has a broad scope and most of the time one subject of specialisation
	Has skills and experience of producing architectures, backed up by appropriate technical skills and experience, including technical breadth
	Responsibility of the architect is the technical content of a project. The Project manager is responsible for the communication and the realising of goals
Business Architect (IT service provider 3)	The translation of developments in IT into useful practices.
	The development of a benefits case for a certain IT investment

Table 6 Architectural tasks combined (taken from Steghuis et al., 2005)

2.4.3 Competences model

Based on the interviews, Steghuis et al. (2005) present three lists of competences: professional competences, intermediary competences, and personality competences. The professional competences are listed per architecture discipline in Table 7. The intermediary and personality competences are listed in Table 8.

General (all architects)	**Business architect**	**Business-ICT architect**	**Software architect**	**Infrastructure architect**
Architectural Principles	Business Administration	*Internal control*	*Applications*	*Frameworks*
Commercial Awareness	Business-IT strategy alignment	Business Administration	*Building and maintaining of IS*	*Tools selection*
Security management	Strategic vision	Business process modelling	Data modelling	*Database Administration*
Possibilities of IT	*Organisational*	Data modelling	Matching Business	Data modelling tools

	Analyse methods		with IT objectives	
Financial justification	Administrative organisation	Matching Business with IT objectives	Methods and Techniques for application and component building	Design and Programming Tools
Innovation	Business process modelling	Business-IT strategy alignment	*Software Process improvement*	Programming Languages
Requirements Engineering	*Organisational design methods*	*Process Simulation*	Change management	Network Management
	Matching Business with IT objectives	Strategic vision	*Requirements management*	Server Management
	Process improvement	Change management	Design and Programming Tools	Network Technology
	Cost/Benefit analysis	Application management	Programming Languages	Server Technology
	Supply Chain Management	Information System Management	*Technical design methods*	*Telephone Technology*
		Methods and Techniques for application and component building	*Technical analysis methods*	*Operating Systems*
		Operational Risk Management		*Technical design methods*
		Organisational Analysis methods		*Technical analysis methods*
		Organisational design methods		*Service Oriented Architecture*
		Cost/Benefit analysis		*Storage technology*
		Portfolio management		*Operational Management*
		Integration		*Middleware*
		Sourcing		*ERP*
		Service Level Management		
		Programme management		

Table 7 Professional competences based on interviews as reported by Steghuis et al. (2005)

Intermediary	**Personality**
Leadership	Persuasiveness
Organisational awareness	Independency
Plan and organize	Persistence

Result drivenness	Decisiveness
Sensitivity and empathy	Initiative
Accurateness	Self Development
Working systematically	Result Driven
Didactical skills	*Innovative*
Listening	*Embracing Challenge*
Negotiation	
Creativity	
Consulting	
Opinion forming	
Teamwork	
Integrity	
Abstraction capacity	
Analytical skills	
Verbal communication skills	
Written communication skills	
Customer Orientation	
Earning trust	

Table 8 Intermediary and personality competences based on interviews as reported by Steghuis et al. (2005)

3 A Framework for IT Architecture

Properly describing an IT architecture requires the description of the architecture from different viewpoints. To ensure the consistency and completeness of the description, a framework of viewpoints is needed. Such a framework is referred to as an *architecture framework*. Typically, an architecture framework defines a number of viewpoints on IT architectures, such as the information viewpoint or the functional viewpoints, and it defines a number of concepts in terms of which to describe architectures, such as the concepts of subject area, service or infrastructure.

IT architect competences are related to IT architecture viewpoints identified in such frameworks, and therefore we will use an IT architecture framework to classify IT architect competences. In this chapter we define the IT architecture framework used in this book.

This poses a problem, for many companies and standardization bodies use their own architecture framework, and all of these frameworks are different from each other. Choosing any one of these frameworks would bias our competence classification towards that framework, and we want to avoid that. Therefore, we have identified a core framework that summarizes the essential elements of a large number of other frameworks. In this chapter, we describe this framework and show how it is related to some of the major well-known IT architecture frameworks. Because the framework is the outcome of a project called GRAAL, we call it the GRAAL framework[5]. The GRAAL framework was first introduced by Wieringa, Blanken, Fokkinga and Grefen (2003). The presentation of the GRAAL framework in this chapter is an updated version of the presentation by Wieringa, van Eck and Krukkert (2005) and the GRAAL Whitepapers[6].

3.1 Systems and emergent properties

We define our framework by taking a systems engineering point of view (Blanchard and Fabrycky 1990, Hall 1962, 1969). The word *system* in this book refers to any coherent collection of elements. Examples are information systems, applications, hardware, a company, the buildings owned by the company, your central heating system and a value network of companies.

Systems have global properties that arise due to the interaction of their parts. For example, each employee of a company on her own cannot produce finished products, but the company as a whole is able to produce finished products; and each separate component of your central heating system cannot warm your house, but the system as a whole can. And each component of an information system is not able to provide the functionality at the level of quality that the information system as a whole can.

[5] http://graal.ewi.utwente.nl.

[6] http://graal.ewi.utwente.nl/whitepaper.php.

These global properties are called *emergent properties,* because they emerge out of the interaction of the parts of the system. Emergent properties may be valuable, as in the examples above, but they may also have negative value. For example, a company may be slow to process orders, or an information system may be hard to maintain. These are properties of the system as a whole too, which are the result of interactions among parts of the system; but they are undesirable. Emergent properties can be desirable or undesirable according to the goals of stakeholders.

3.2 System architecture

It is the job of an IT architect to maximize the number of desirable emergent properties and minimize the number of undesirable ones. In other words, the architect tries to create synergy among the parts of a system. We define the *architecture* of a system as the structure by which its desired properties emerge. The difference between a house and a pile of bricks is that the house has an architecture; and it is this architecture that creates desirable global properties of the house, such as that it offers places to live and sleep, that it shelters its inhabitants from weather conditions, and that it creates an atmosphere in which the inhabitants feel at home. To turn a pile of bricks into a house we put them together to realize a structure, and this structure has an architecture to the extent that it creates emergent properties that agree with the goals of the inhabitants of the house.

Thus, the architecture of a system is not only the structure of the house; it includes the way in which the structural elements interact to create the desirable overall properties of the system. Architecture is the link between the structural elements of a system and the goals of the stakeholders of the system. We can summarize this by the slogan that architecture is structure plus synergy.

In the same way, the architecture of a software system is not only its structure; it is also the way in which its structure creates desired overall properties: services, behaviour, interfaces, reliability, usability, etc. The architecture of a business is the way its parts work together, the structure in which they are put, that makes the business as a whole able to produce products and service and have certain quality properties such as reliability or customer-friendliness.

How many architectures does a system have? Some software engineers say that systems have many architectures, e.g. a module architecture, a run time architecture, a call graph, etc. Alternatively, other people say that a system has *one* architecture but that there are many views on it. For example, a house has one architecture but there are different views on it, showing the view of the builder, the electrician, the plumber, and so on. These are just manners of talking, and we think there is no fundamental difference between them. In this book, we will choose the second approach, i.e. we will talk about “the” architecture of a system but admit that the architecture of a system is usually very complex and needs to be described from many different viewpoints. It is the job of an architecture framework to define these viewpoints.

3.3 Architecture views

The different viewpoints defined by an architecture framework are ways to master complexity. The simplest way to master complexity is to omit details in a description of a system. This gives us the first dimension of the GRAAL architecture framework, which we call refinement:

– The *refinement levels* of a system description differ in the amount of detail included in the description.

Refinement levels are levels of system *description*. The three dimensions defined next concern the semantics of system description. These three dimensions are generally recognized in systems engineering.

– *Aspects* of a system are externally observable properties of the system.
– The *composition* of a system is the set of its parts and their relationships.
– The *phases* of a system are the different stages in its life.

Each of these views is a way of mastering complexity. We can master complexity by considering only one aspect of a system, or by zooming in on a subsystem, or by considering the system only in one phase of its life. Or we can do all of this at the same time: Zooming in on one aspect of one subsystem in one phase of its life. And we can do any of this at any level of refinement. This gives us a very powerful set of complexity-reduction techniques, which we now describe in more detail, beginning with the three semantic dimensions.

3.3.1 System aspects

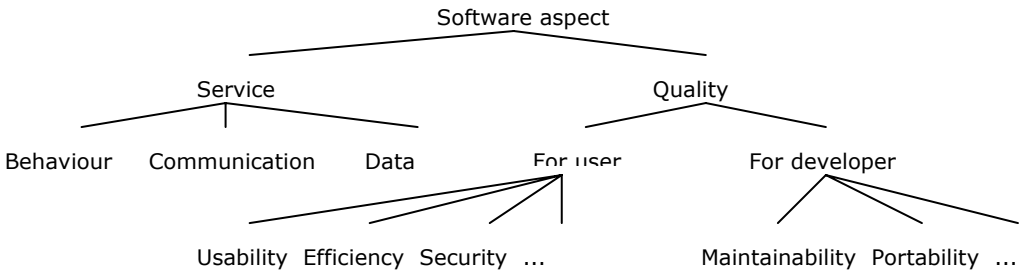


Figure 6 Some software aspects

Figure 1 shows a classification of aspects of the kinds of systems we are interested in, namely businesses and IT systems. We take a service-oriented view, which means that we restrict our attention to the services performed by the system for its environment, and ignore other aspects such as the delivery of goods to the environment.

A *service* of a system consists of interaction with actors in the environment of the system, such as users, customers or other software, which is performed at a certain *quality level*. For example, an information system provides information to its users (a service) with a certain reliability, currency and accuracy (quality attributes). The business as a whole offers services to its environment, such as financial services or logistics services,

and it does this at a certain quality level. Well-known software quality attributes are usability, efficiency and reliability for users, and maintainability for developers. Important business quality levels are reliability, responsiveness and availability. Software quality and business quality are in many cases related. Figure 6 shows a number of software quality attributes.

A software service is a meaningful interaction between the software system and its environment, and is therefore characterized by three functional properties (Figure 6).

- *Behaviour*. The interactions of a service are performed in a sequence, and they contain choices.
- *Communication*. The interactions consist of communications with other actors, such as people, devices, businesses, and software.
- *Data*. The interactions consist of data exchanges, i.e. meaningful messages that are communicated with the environment.

Software development methods offer various notations to represent these aspects. For example, event lists and state transition diagrams (statecharts) can be used to represent behaviour. Data flow diagrams and use case diagrams can be used to represent communication between processes and actors. Entity-relationship diagrams can be used to represent the semantics of data. These are just examples; there are many other notations available.

Software services are performed at certain quality levels, of which Figure 6 lists a few. It is convenient to classify quality attributes according to the actor that experiences the quality, such as users, customers, developers or maintenance personnel. Business services also consist of behaviour in which data is exchanged with the environment of the business at a certain quality level. But in business services, this data must provide information or knowledge to the customers of the business.

3.3.2 System composition

A second mechanism to master complexity is to consider one subsystem only. Any system is part of an aggregation hierarchy, as shown in Figure 7.

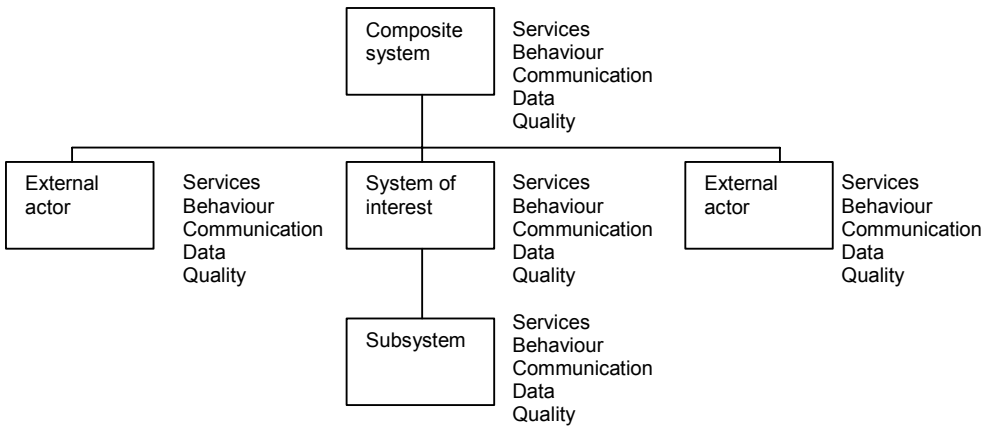


Figure 7 System aspects occur at every level of a decomposition hierarchy

For example, a business (composite system) consists of organisational units, which contain software systems, which contain software modules, which contain software objects, etc. At each level, systems have quality aspects. For example, software objects offer services with a certain quality of service, and each service has behaviour, communicates with other objects, and exchanges data. The same can be said of modules, of subsystems, of entire systems, and of systems of software systems, of organisational units, of businesses, and of constellations of businesses. It is an important job of an IT architect to relate services and quality attributes of the parts of a system to the overall services and quality of the entire system.

3.3.3 Composition in three worlds

In the real world, this simple picture gets more complicated because it is not always simple to decide what is a part of what. To understand this, we must distinguish three worlds:

- The *physical world* is the world of computers, cables, printers, wireless access points, and in general anything that can be described using the basic measuring units of physics, Meters, Kilograms, Seconds, and Amperes. Entities in our physical world usually make noise, generate heat, and can be dropped on the floor.
- The *social world* consists of roles people play, and of organisations, departments, money, responsibilities, business processes, markets, customers and suppliers, and in general the processes and structures defined by human institutions.
- The *software world* consists of software applications, computerized information systems, office software, ERP systems, workflow management systems, database management systems, middleware, operating systems, assembly language programs and even of micro-programs running on computers.

Ultimately, software is a state of a physical computers but software has different kinds of properties than hardware. For example, software can be copied at virtually no cost. The social world too is ultimately realized in the physical world of buildings, roads and human bodies, but it has very different properties than the physical world. For example, in the social world a department can move from one organisation to another without changing its physical location.

The conceptual confusion about decomposition arises from the fact that in the physical, social and software worlds, decomposition has a quite different meaning.

- For example, a physical computer is composed of many physical components. This means that each component is physically smaller than the computer, and is located inside the computer. The component plays a role in the service that the computer delivers to its environment.
- A business is composed of many departments, but this does not mean that the department is physically “smaller” than the business. Departments and businesses are legal constructions and they cannot be described using the measurement units of physics. Rather, being part of a business means having a certain legal relationship to the business. In particular, the department plays a role in the provision of services by the business to its environment.

– A software system may be composed in many ways; for example it can be composed of modules. Again, software is not physical: Software is in the proverbial holes in the punched cards, and software has no weight. Rather, a module is part of a larger software system because it is part of the logic of the software system. In particular, it plays a role in the services that the software system provides to its environment.

All in all, there are two elements of the meaning of decomposition that appear in all three worlds.
– The composite system *encapsula*tes its parts. To interact with the part, you have to pass through the interface of the composite system. In the physical world this means that the part is inside the composite, but in the social and software worlds, this means that to interact with the part you have to interact with the composite system.
– A part of a composite system *provides a service* to the composite system, by which the composite system itself is able to provide its services to its environment.

If we remove a part of a system and place it in the environment, providing its services to the system as well as to other systems, we have introduced a layering structure, discussed next.

3.3.4 Layering

In the social world and software world, it is relatively easy to remove part of a system and place it in the environment. For example, a company A can decide to turn the logistics department into an independent company that still provides logistics services to A but can now also offer them to other companies. And a software engineer can decide to remove a module from a software system by making its services available to other software systems too. We then remove the encapsulation of a component but preserve its service provision. This turns a decomposition relationship into a *layering* relationship.

All IT architecture frameworks recognize layering as an important structuring mechanism. Figure 8 shows the layers identified in GRAAL, and relates these to the three worlds. In general, systems at each layer provide services to systems at any higher layer.

Social	Business environment	Customers
	Business	Employees, processes, ...
Software	Enterprise software systems	Applications, information systems
	Software infrastructure	Operating system, middleware, servers, ...
Physical	Physical infrastructure	Buildings, machines, cables, roads, ...

Figure 8 Layers of GRAAL

The layers are the same as the ones used in the classification of architecture disciplines in Section 2.1:

- Business environment: entities in the environment of the organisation to which the organisation delivers products and/or services. For commercial companies, the most important type of elements of the business environment are their customers.
- Business: the products and services that the organisation produces for its environment, the processes that create these products and services, the employees who perform those processes, the formal and informal relations between those employees, etc.
- Enterprise software systems: organisation-specific software systems that support the processes and people in the business, such as administrative systems, process support, and decision support systems.
- Software infrastructure: software systems that are not specific for the organisation, such as operating systems, database management systems, email servers, etc.
- Physical infrastructure: processors, disks, network routers, switches and cables, and all other physical objects that are needed to run the software systems that constitute the business systems and software infrastructure layers.

Layering adds considerable conceptual power to architecture frameworks. For example, in Figure 9 we combine our layering structure with decomposition and aspects.

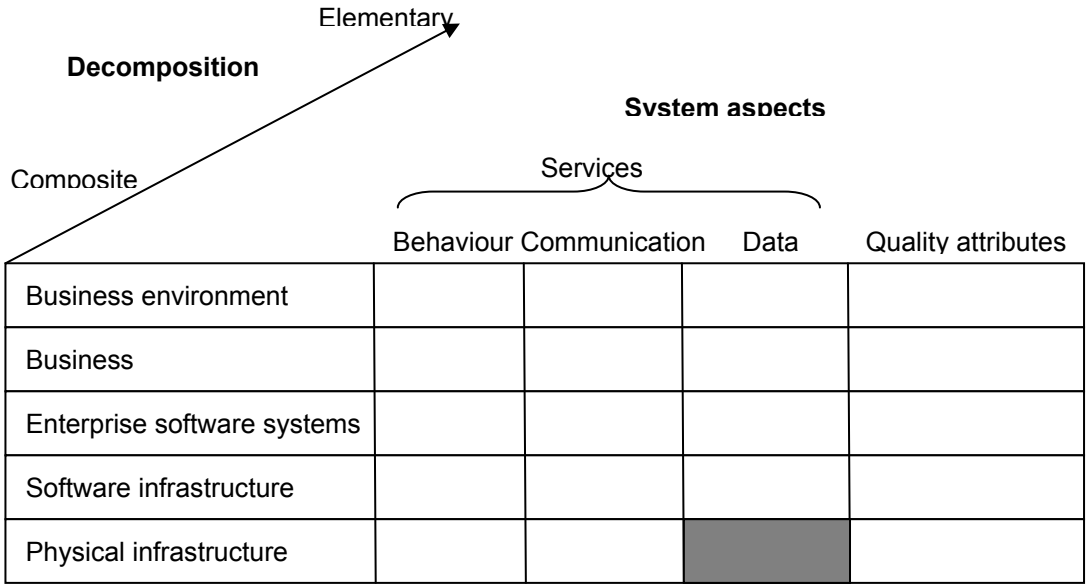


Figure 9 Combining layering with decomposition and aspects

Figure 9 shows that systems at each layer in our framework provide services that consist of behaviour, communication and data, except in the physical world, where the concept of data is not defined. Physical systems have behaviour and interact with their environment, but as soon as we recognize data, we have passed to the software world in which bits are manipulated. Systems in a software infrastructure pass data between each other, and enterprise software systems store and manipulate data that is of importance to a business. The business provides services to its environment in which data is exchanged with its environment; depending on the quality of the service provides, this data represents information or knowledge for the business customers.

Note that there is no information system layer in this framework. Rather, data is a column in this framework, because the data aspect is present in *all* software and social layers of the framework, not just in one layer. The enterprise software systems layer contains systems contains systems with information provision functionality as well as applications such as decision support systems, process support systems and personal productivity tools.

Systems at each layer have an internal structure of components. Along the decomposition dimension systems encapsulate their components. Each of these components itself provides services to its own environment, consisting of behaviour, communication and data offered at a certain quality of service. Layer, aspects and decomposition together offer powerful complexity reduction techniques for understanding systems.

3.3.5 System phases

There are two more ways to master complexity of a system, each of which provides us with one additional dimension of our architecture framework. One way to master system complexity is to consider only one stage in the life of a system. Any system goes from conception through acquisition, use and maintenance to disposal.

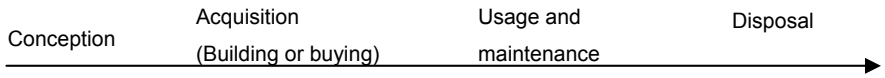


Figure 10 Phases in the life of a system

An important part of the problem of aligning software to the business is coordination of future development of software systems. Of every system, several versions may exist. Many systems are supplied by third parties, each with their own release frequency. Coordinating all of this is a major problem in practice, and identifying the stages in the life of a software system helps mastering this complexity.

3.3.6 Description refinement

Aspects, decomposition and layering, and phases are semantic ways to master complexity: They are structures of systems, which we can use to structure our descriptions of these systems. The final way to cope with system complexity is to omit details from a system description, an operation we call “abstraction”. We can do this at any layer in our framework. Figure 11 shows some illustrative refinement relations.

	Abstract	Description refinement → Detailed
Business	Market, Distribution channel,	Event, Communication channel,
Software	System mission, infrastructure standard	System transaction, Software library
Physical	Geographic area	Network topology, Machine

Figure 11 Levels of description refinement

Not only can we describe systems at each service layer at any level of detail, we can do the same with systems at different levels in the decomposition hierarchy. For example, we can describe a software system very abstractly by describing its mission and responsibilities, or very detailed by describing all its transactions and the structure of the data input and output by the system. And we can describe a software object by describing its mission and responsibilities, or in a very detailed manner by describing the syntax and semantics of its operation calls, and the communication protocol to be used when calling an operation. We can represent this in a rectangle called the magic square (Harel and Pnueli 1985).

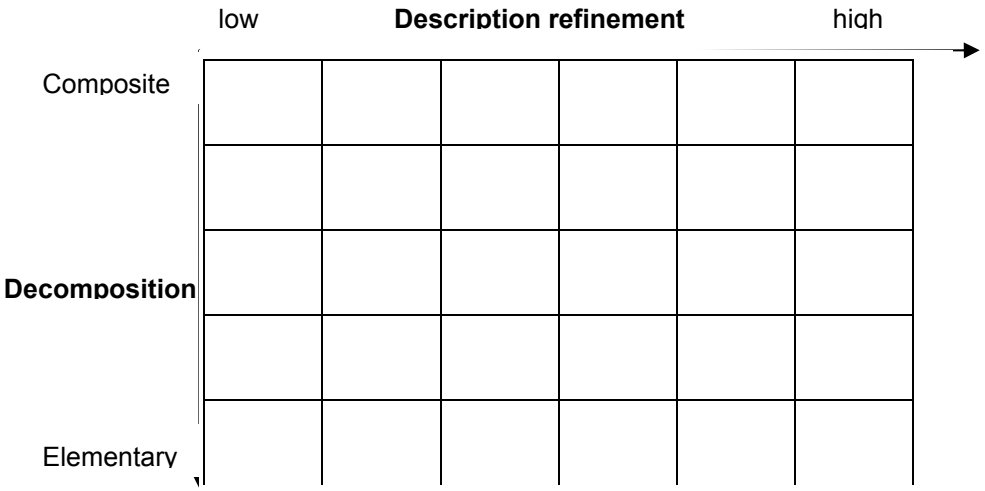


Figure 12 The magic square

The square represents decomposition levels and refinement levels of one system. For example, the square in Figure 12 can represent the entire enterprise software system layer of a company. In the vertical dimension, this layer can be decomposed in applications and databases, and each of these can be decomposed in one or more software components, these components can be decomposed into, say, modules, which can be decomposed into

objects. At each of these decomposition levels we can describe what we see there at many levels of refinement, from highly abstract (few details) to very refined (many details).

For example, we can describe the entire application layer by saying that it provides information processing support for the sales and logistics department (few details) but we can also elaborate this into a very detailed description of the services provided by this layer: Order administration, route planning, etc. These descriptions can be given without referring to any of the software systems in the application layer, but only to the services provided by these systems.

Moving a few decomposition layers down the square, we can describe a route planner by saying that it should provide optimal route plans for vehicles based on their current location, capacity and destination of goods. We can then refine this description by adding details about how the data about vehicle positions is kept up to date, how the system interfaces with the order system, etc.

Decomposing a system into components is not the same thing as moving to a more detailed description level. In general, we can move horizontally through the square by adding detail without adding information about components, and we can move vertically by adding information about components but stating at the same level of refinement. Or we can do both at the same time: describe a component, and do this at a higher level of refinement (more detail). What makes the square magic is that it should not matter for the final result by which route we arrive there.

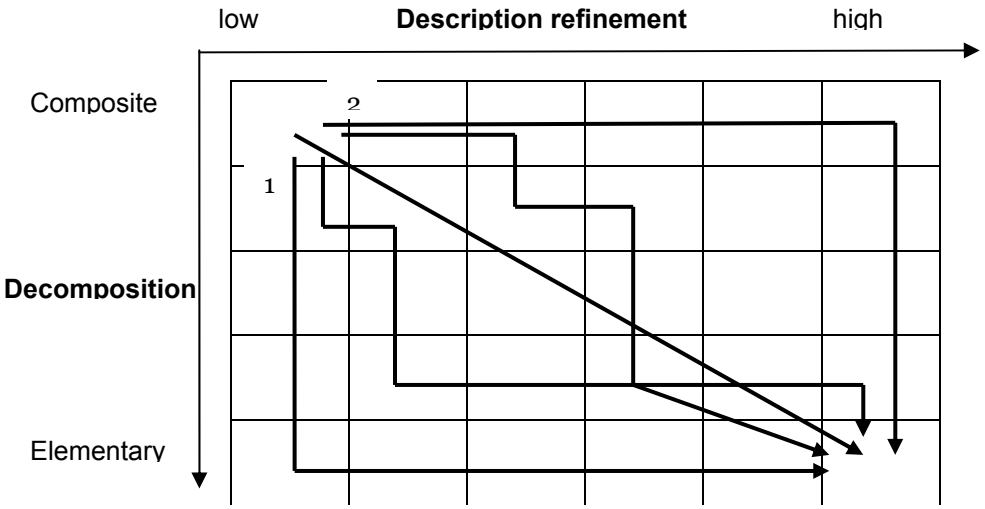


Figure 13 Different routes to arrive at the same low-level system description

At the end of the day, all lowest level components of the entire software layer are described at a very high level of detail. The descriptions may be scattered over the company and its software suppliers, and they may be buried deep inside the code or even consist of parts of the code. No person would be able to comprehend all of it. But each description can be allocated to a cell in the magic square, because it describes a certain component (vertical position) at a certain level of detail (horizontal position). And if an architect is developing an architecture, she can describe all composition levels at a high level of abstraction, after which each component can be developed by adding more detail to the de-

scription (route 1 in Figure 13) or she can describe the entire layer at a high level of detail, followed by an implementation of this in software systems and their components (route 2), or she can follow any path in between these extremes.

The same is true for the other layers of the GRAAL framework. For example, we can describe the mission and external services of a business at a high abstract level and then elaborate this into detailed descriptions of relevant events and transactions with the environment without ever referring to people or software executing these transactions. These detailed descriptions can then be mapped to tasks of low-level components of the business. We can alternatively describe all people and software in the business at a high level of abstraction, describing the missions and major responsibilities only, and then elaborate each of these descriptions with more details. In both alternatives, we end up with detailed descriptions of low-level components in the lower-right corner of the magic square, but we arrive there by different routes.

3.4 The GRAAL framework

The GRAAL framework is defined by Figure 9 to Figure 11. We have provided more explanation of the framework in earlier publications (Van Eck, Blanken and Wieringa 2004). In the next chapters we will use the framework to classify architect competences. In the remainder of this section we will show how other well-known frameworks map to the GRAAL framework. This shows that the GRAAL framework can be used as a common denominator of the other frameworks.

3.5 Comparison with other frameworks

3.5.1 Zachman

Zachman presented his architecture framework in 1987 and extended it together with John Sowa in 1992 (Zachman 1987, Sowa and Zachman 1992). In the extended framework, each system can be described from six points of view, namely data, function, network, people, time and motivation. The data, function, network and time viewpoints correspond with the GRAAL aspects of data, service, communication and behaviour (Figure 14). The people and motivation viewpoints are two aspects of the business in which the software is embedded. People are part of the business and therefore of the business aggregation hierarchy. Motivations are part of high-level mission statements at any level in the business decomposition hierarchy. It is more accurate to speak of goals instead of motivations: The business as a whole has goals, each of its departments has goals and employees have goals too; and customers have goals. Note that the quality aspects are absent from the Zachman framework.

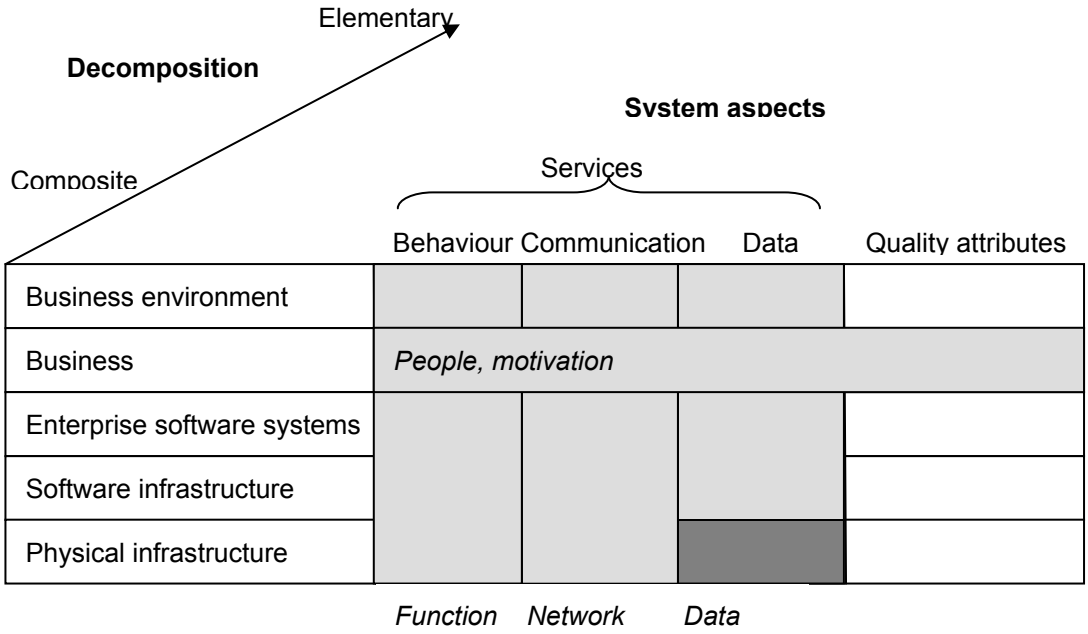


Figure 14 Zachman's framework mapped to GRAAL

Descriptions from each of Zachman and Sowa's points of view can be given for different stakeholders, who have different interests. Zachman and Sowa distinguish descriptions for the planner, owner, designer, builder and subcontractor. The GRAAL framework does not distinguish these stakeholders. Nor does any other framework identify these stakeholders. The reason for this is probably that different development processes would involve different kinds of stakeholders and an architecture framework should not restrict itself to any one of these processes and stakeholders.

The GRAAL framework distinguishes levels of refinement and decomposition in addition to the different aspects of a system. Jointly, these dimensions are sufficient to define descriptions relevant to these different stakeholders.

3.5.2 The four-domain architecture

Iyer and Gottlieb (2004) propose a pragmatic improvement on Zachman and Sowa's framework by reducing the number of relevant viewpoints to four: the process domain, information domain, infrastructure domain and organisation domain, and ignoring the stakeholders identified by Zachman and Sowa. The resulting four-domain framework corresponds to the following cells of the GRAAL service provision layers.

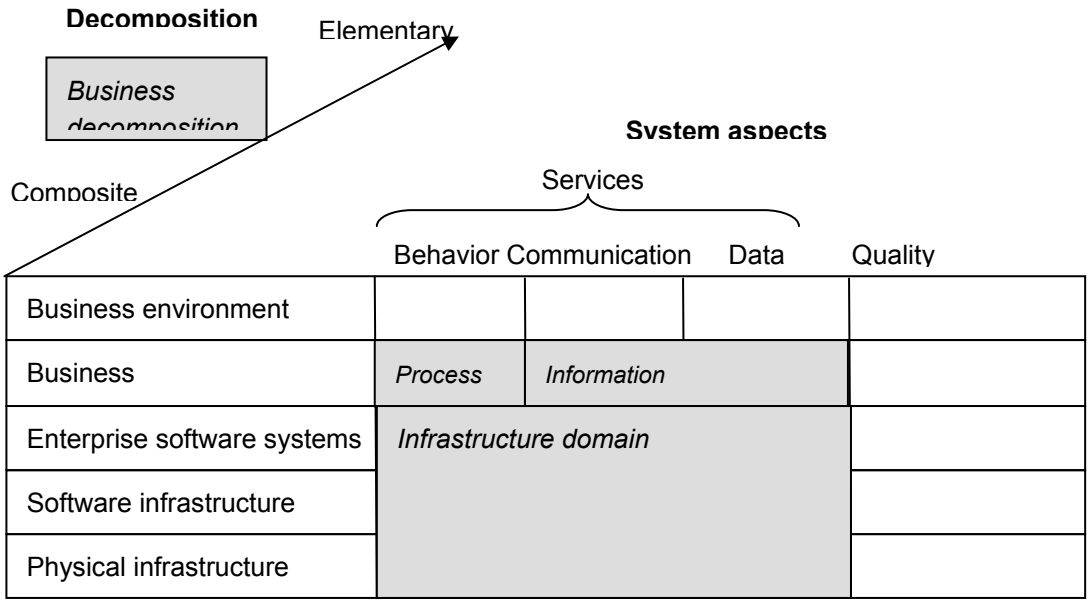


Figure 15 The four-domain framework mapped to GRAAL

Compared to the GRAAL framework, the four-domain architecture lumps all software and hardware infrastructure under the general term 'infrastructure'. The organisation domain consists of the roles and departments of the business, which corresponds to the decomposition dimension at the business level (Figure 15). The business process domain considers behaviour at the business level, and the information domain considers information flows and information meaning at the business level.

3.5.3 ArchiMate

ArchiMate provides a set of concepts and a modelling notation for representing enterprise architectures (Jonkers, Lankhorst, van Buuren, Hoppenbrouwers, Bonsangue and van der Torre 2004). The concepts and the elements of the notation can be placed in a two-dimensional framework, which ArchiMate calls the architectural framework. The two dimensions are an aspect dimension (with three aspects: information, behaviour and structure) and a dimension that distinguishes three layers: the business layer, application layer and technology layer. The aspect dimension is very similar to the aspect dimension of GRAAL. The layer dimension of ArchiMate corresponds to the service provisioning dimension of GRAAL (

Figure 16).

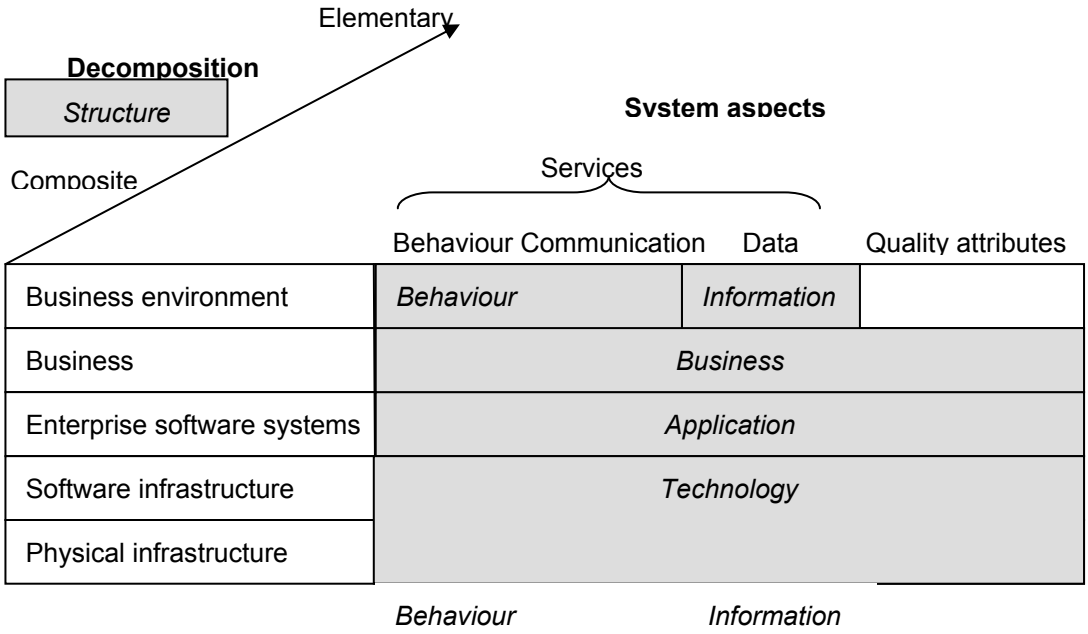


Figure 16 ArchiMate mapped to GRAAL

3.5.4 The 4+1 model

Kruchten's 4+1 model (Kruchten 1995) defines the logical and process views of a software system, which correspond roughly with our decomposition dimension and behaviour view, respectively. Furthermore, Kruchten defines a physical and development view, which correspond roughly to our infrastructure layer and to our system phases dimension, respectively.

3.5.5 Henderson and Venkatraman, Maes, and IAF

Henderson and Venkatraman (1993) proposed a framework for strategic alignment that distinguishes two dimensions: functional integration of business and IT, and fit between strategic and operational levels of a business. This can be mapped to two dimensions of the GRAAL framework as shown in Figure 17.

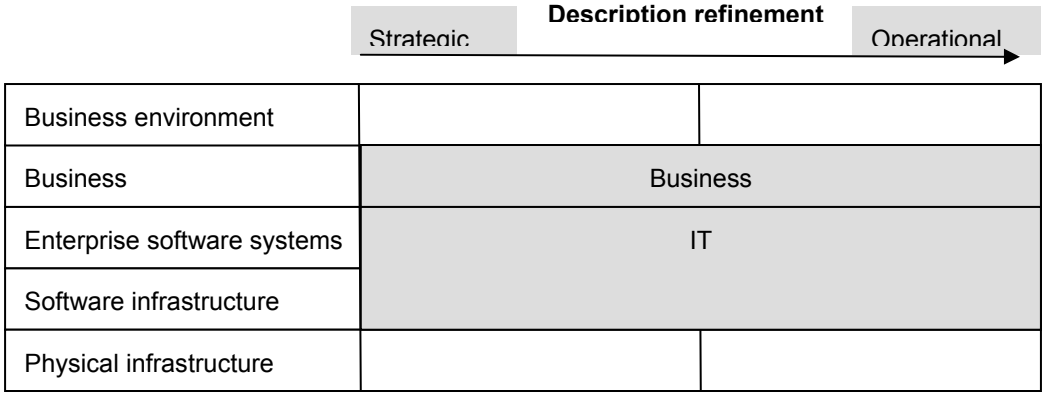


Figure 17 Mapping the strategic alignment model to GRAAL

The strategic fit dimension does not quite map to our description refinement dimension, since strategic fit involves more than varying the level of detail in one's description.

Maes (1999) extended the framework of Henderson and Venkatraman, keeping the same dimensions, but distinguishing three instead of two points on each dimension. The nine cells of Maes's framework can be placed in the GRAAL framework too (Van Eck, Blanken and Wieringa 2004).

Maes' extension of the Henderson and Venkatraman framework was extended again to create the Unified Architecture Framework (Maes, Rijsenbrij, Truijens and Goedvolk 2000). UAF splits the technology layer of the first (1999) Maes' extension in two layers, called Information Systems and Technology Infrastructure. Moreover, UAF adds two dimensions to the (2D) Maes / Henderson-Venkatraman framework: a dimension that distinguishes five design phases and a dimension that distinguish specific viewpoints such as security and governance. These two dimensions have been taken from Capgemini's Integrated Architecture Framework (IAF); the UAF is as such a merger of IAF with (Maes' extension of) the Strategic Alignment Model of Henderson and Venkatraman. The design phases dimension is comparable to the GRAAL lifecycle dimension. The viewpoint dimension of IAF is defined only in terms of example viewpoints. The viewpoints given can be incorporated in the aspect dimension of GRAAL.

3.5.6 TOGAF

The Open Group Architecture Framework (TOGAF) distinguishes the business architecture, information system architecture, and technology architecture, which correspond to the business layer, business system layer, and infrastructure layer of the GRAAL framework.

4 Architecture Disciplines

4.1 Basic architecture disciplines

It is the task of an architect to design systems that have the emergent properties desired by their stakeholders. Architects can work in the physical, software or social worlds, and often must align two of these worlds to each other. This gives us a number of different architecture disciplines, located at different places in our GRAAL framework. Consider the layers of this framework, rearranged in Figure 18.

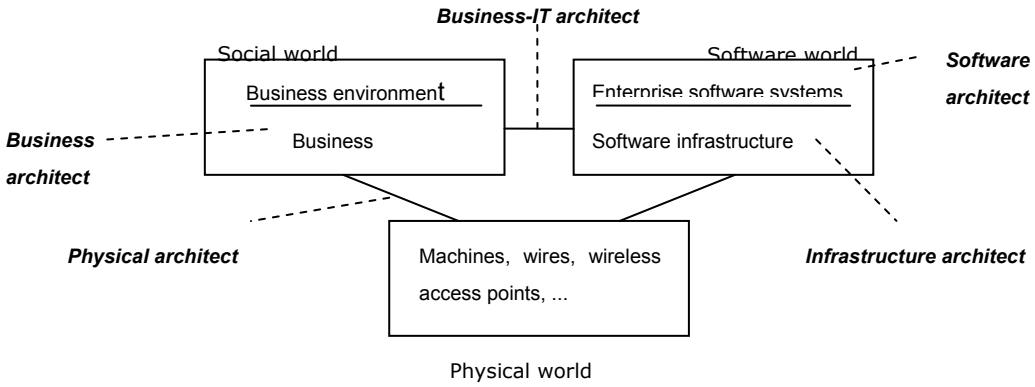


Figure 18 Some architecture disciplines

Figure 18 shows the three worlds of GRAAL as separate blocks, with their internal layering structure, and their alignment relationships. The diagram indicates that a physical architect designs physical structures, such as buildings, to satisfy social business needs, match business processes, or in one phrase to satisfy social requirements. A business-IT architect does the same but now for the software world. A business-IT architect analyzes goals and needs in the social world and designs structures in the software world that, if implemented, would meet these goals and needs.

Usually, the architect only designs at the grand scale. A building architect designs an overall building structure and appearance that would fit the identified needs, and a business-IT architect designs a collection of enterprise systems that supports the needs of the business.

However, note that these two architects can work in both ways: Ensuring that physical structures or software structures match social requirements, but also ensuring that social expectations match the designed physical or software structures.

More in detail, the different architecture disciplines listed in figure 18 consist of the following tasks.

– The **business architect** analyzes business problems and goals, and designs business solutions. These solutions may involve now tasks, processes, roles, departments, structures etc. to be realized in the business. Business architects can work at any aggregation

level: Some business architects design an entire business; others design a single workplace.

- The **business-IT architect** analyzes business problems and goals and designs IT solutions. Depending on the aggregation level at which he or she works, the business-IT architect produces an overall architecture of a collection of enterprise systems that would meet the business needs, or he or she produces a specification of a single enterprise system in context. Some business-IT architects analyze the needs of an entire business and design an entire application layer; others analyze the needs of a particular user group and design one application as solution.
- The **physical architect** takes care of alignment between the social world and the physical world. Some physical architects design cities; other design buildings.
- The **software architect** analyzes a software specification produced by a business-IT architect and designs a software structure that will implement this specification. Some software architects design complex distributed applications; others design a single module.
- The **infrastructure architect** analyzes problems and goals for the software infrastructure. He or she estimates the need for infrastructure resources by business systems, analyzes business problems and goals, and designs an infrastructure that meets those needs and goals. Infrastructure architects usually also deal with alignment with hardware. Infrastructure architecting is a demanding task, for infrastructure architects therefore consider business systems, the business itself, as well as hardware. Some infrastructure architects design a new architecture for the entire infrastructure layer; others design the architecture of workflow management support.

Note that we use the term "design" here in its general sense of making a plan of what to build. In this general sense one can design a business, a job, a business process, or a software system. It is the task of an architect to design an assembly of solution elements that would fit stakeholder goals. For a physical architect, the solution elements consist of walls, doors etc. and for a business architect, the solution elements are job roles, processes, decision procedures etc.

4.2 Frequently occurring disciplines in practice

Different companies distinguish different architecture disciplines and use different names for them. In the survey of architecture disciplines recognized in seven companies presented in Section 2.4, we found the names listed in Table 9. We compare these names with the ones used in GRAAL and in the NGI framework (discussed in section 2.1).

GRAAL	NGI	A	B	C	D	E	F	G
Businessarchitect			Enterprise architect	Enterprise architect	Enterprise architect		Business architect	
			Systems architect: Business					
Business-IT architect	Information architect	Information architect	Systems architect: information				Information-architect	
	Data archi-							

	tect							
		IT-business architect						
Soft-warear-chitect	Software architect		Systems architect: information systems,	Solution architect	Application architect	Application architect	Information system architect (application architect, database architect)	Application architect
					Integration architect			Technical application architect
Soft-ware infra-struc-ture architect	Technical infrastruc-ture archi-tect	IT archi-tect	Software architect	Techni-cal archi-tect	Infrastruc-ture archi-tect	Infrastruc-ture archi-tect (devel-opment a., technical a.)	Information technology architect	
	Network architect							
Physical architect			Systems architect: technology infra-structure					

Table 9 Architecture disciplines in GRAAL, NGI, and in seven companies (taken from Voermans, Steghuis and Wieringa 2005)

Table 9 shows that different companies use different classifications and names of architect roles, but that there is a large degree of similarity and that this similarity is captured by our framework. To explain the mapping of these names to the GRAAL disciplines we list them as subdisciplines in Figure 19.

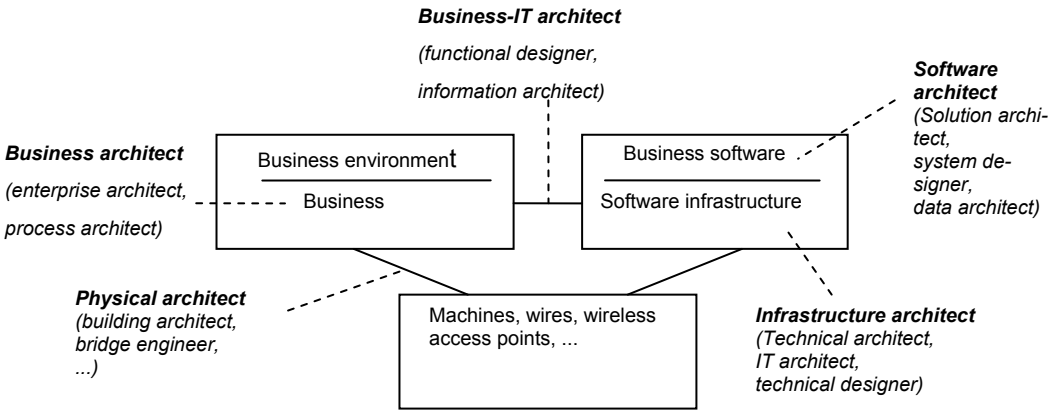


Figure 19 GRAAL architecture subdisciplines

The meaning of these disciplines is as follows.

- Business architect:
  - The ***enterprise architect*** does much the same as a business architect, but usually at high aggregation level. The enterprise architect does not design individual systems but looks at the entire enterprise system layer.

  - The ***process architect*** does the same but focuses on business processes as a solution. In terms of the GRAAL framework, he or she focuses on the behaviour aspect in the business world.
- Business-IT architect:
  - The ***functional designer*** focuses on the desired functions the software solution. In terms of the GRAAL framework, he or she focuses on the services to be offered by the software solution.
  - The ***information analyst*** analyzes information needs in the business, and the ***information architect*** designs information flows and structures as a solution to business problems. In terms of the GRAAL framework, these two roles focus on the information aspect of the business world and software world.
- Software architect:
  - Some companies call software architect ***solution architects.*** This is misleading because software may be a problem to be solved rather than a solution to be implemented.
  - A ***system designer*** does the same as a software architect but on a less grand scale.
  - The ***data architect*** designs data structures.
- Infrastructure architect:
  - Infrastructure architects are also called ***technical architects.*** This is misleading, for there are other things technical besides software and the hardware on which it runs.
  - Yet another name of infrastructure architects is ***IT architect.*** This is misleading too, for there are other IT systems besides infrastructure systems.
  - A ***technical designer*** does the same thing as an infrastructure architect but on a less grand scale.

The above definitions are approximate and the company manuals defining job roles usually provide a lot more detail. Nevertheless, the above mapping to the GRAAL disciplines should make it possible to compare architecture disciplines across companies and across standards.

5 A framework for competences

We have now given a framework for architecture, and identified different architecture disciplines. In order to classify architect competences we must also offer a framework for competences. We do so in this chapter, using some concepts from learning theory.

5.1 Competences and proficiency levels

Competences are different from knowledge, skills or attitudes. A competence is the ability to respond adequately in a concrete situation (Ten Dam and Vermunt 2003). The response will be based on knowledge, skills and attitude, but more is needed to acquire competence. For example, one may be able to drive a car (skill), know the traffic rules (knowledge), and behave in a careful manner (attitude), and still be a lousy driver in some concrete situations. Having the required knowledge, skills and attitude is not enough.

Moreover, competences are related to concrete situations. A competent city driver shows her competence by responding adequately in city traffic, but the same person may be incompetent for driving long distances. Similarly, someone may be a competent enterprise architect but an incompetent data architect or vice versa.

These examples should make clear that competence is not something learned in school but something exhibited in practice. In relation to schooling it is useful to distinguish three experience levels that mark the road from student to professional.

- **Learned.** The knowledge required for a competence has been learned, and the student passed an exam to test this. For example, the student can understand process models and architecture diagrams.
- **Applied to examples.** The student has applied the knowledge and used the required skills under supervision. For example, the student has made process models or architecture diagrams for artificial cases.
- **Used in practice.** The professional has applied knowledge, skills and attitude independently in concrete situations.

Within the category of professional use, we should make three distinctions based on the level of understanding.

- **Ability.** The architect responds adequately in concrete situations, but cannot explain why one alternative was chosen above another. For example, she has made adequate process models in a particular project, but cannot explain why certain modelling choices have been made.
- **Accountability.** The architect responds adequately in concrete situations, and can explain why one alternative was chosen above another. For example, she has made adequate process models in a particular project, and can explain why certain modelling choices have been made.

– **Reflection.** The architect responds adequately in concrete situations, can account for them, and is able to describe her own performance, and the suitability of the tools and techniques used, and can propose improvements to any of this.

It is possible that a talented student has learned but not yet applied a technique but nevertheless can reflect on her performance and on the properties of the technique. However, for our purposes we will apply the levels of understanding only to experienced professionals. This gives us the following matrix of proficiency levels.

		Understanding		
Experience		**Ability**	**Accountability**	**Reflection**
	Learned	Conceptual		
	Applied to examples	Beginner		
	Used in practice	Experienced	Advanced	Expert

Table 10 Proficiency levels

Most companies that we investigated in the survey introduced in Section 2.4 distinguish proficiency levels from conceptual to expert similar to those listed in Table 10. A student fresh from school with architecture knowledge and skills should have proficiency level Beginner. After some years of experience she should be Advanced. Depending on personality properties such as communicativeness and intelligence, the architect can progress to expert level. We should repeat that this level must be related to a specific class of concrete situations. An architect with expert competence in one discipline can have a beginner's competence in another.

5.2 The competence pyramid

IT architects in user companies—banks, insurance companies, industries, government organisations—may have a background in 'the business' or in IT. Apparently, the missing knowledge, in IT or in the business domain, can be acquired. Apparently, the crucial competences of IT architects are not technical. This can be explained by means of the iceberg structure of competences (Bergenhenegouwen, Mooijman and Tillema 1999).

At the top of the iceberg, we find the observable **professional competences** required to exercise a profession. For example, an IT architect should have knowledge of the business domain as well as the IT domain, and be able to design an IT architecture that fits business strategy. Professionals acquire this knowledge and these skills by schooling and on the job, and their presence can relatively easily be observed.

Lower down the iceberg, we find more knowledge, skills and attitudes, that are however increasingly less teachable and observable the lower we go. **General skills** that can be used in almost all professions include the following:

– Social and communicative skills

– General technical knowledge
– Management skills
– Problem-solving skills

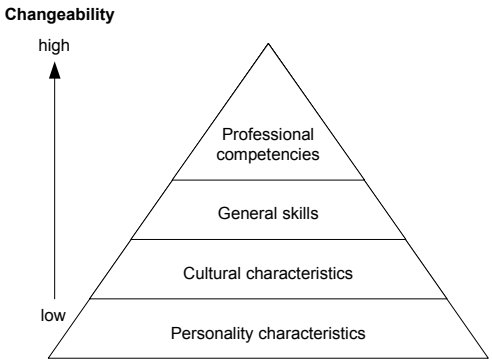


Figure 20 The competence iceberg (based on Bergenhenegouwenet al. 1999)

For example, an IT architect should be able to listen to people, to communicate technical knowledge to non-technical people, she should understand general principles of systems engineering, be able to apply change management skills and be able to identify and structure complex problems. All of this can be learned to a certain extent, although personality characteristics play a role in the ability of a professional to learn these skills. Even if the skills could be learned by anyone, independent of his or her personality, they are soft skills that require maturity to learn and apply. Their presence can less easily be tested as technical knowledge and skills. Skill like these are useful not only for IT architects but for almost any professional.

Deeper down in the iceberg we find **cultural characteristics** that are learnt not by schooling but by becoming a member of a community. Examples of these are the following:

– Professional ethics
– Cultural values of the business, such as customer orientation
– General norms and values

Cultural characteristics like these define “who we are” and “how we do things around here”, and they also define part of the personal identity of a professional. They are learned by a slow socialization process and they are not easily described explicitly. Observing their presence or absence likewise takes a long time, and it is hard to give crisp and clear criteria for their presence.

The lowest layer of the iceberg consists of **personality characteristics** such as communicativeness, empathy, intelligence and emotional stability. These characteristics cannot be taught, not are they acquired by socialization, but they can be stimulated or inhibited when present. For example, a leadership course does not teach leadership skills in the same way as the latest techniques in aspect-oriented programming would be taught. Rather, a leadership course makes participants aware of their leadership characteristics and teaches them how to use and improve them if they are present.

The iceberg contains knowledge and skills (the top two layers), and attitudes (the bottom two layers), but not competences. A competence is the ability to respond adequately

to a concrete situation *based on* knowledge, skills and attitudes. The iceberg classifies knowledge, skills and attitudes into those that are more easily learnt by the professional and observable by others, and those that are less easily learnt by the professional and observable by others. The personal and cultural characteristics are the most important ones to evaluate when selecting personnel, because they are the hardest to change.

6 IT architect competences

What are the competences of a good IT architect? To answer this question as objectively as possible, we should define and validate observable criteria for good architects, and then study these architects to find out which competences they have. Next, we should show that these competences are what make these architects *good* architects. This is a tall order, and it is questionable whether it can be done at all due to the many variables that play a role in answering these questions. What we present in this chapter is more modest and more doable: We analyzed competence profiles of a number of IT-intensive companies, interviewed senior IT architects at those companies, and conducted a survey among IT architects visiting the Dutch National IT Architect Conference (*Landelijk Architectuurcongres*) in 2004. In this chapter we present a summary of the results of this research.

6.1 Professional competences

Professional competences concern the domain of expertise of the IT architect, such as infrastructure, enterprise software or business domains. These are called 'professional competences' because they require specialist training in specific analysis and design pertaining to the domain of expertise of the IT architect. Note that we include business analysis and design competences under this heading, as well as expertise concerning strategic business-IT alignment. All of these architecture disciplines require specialist knowledge and skills. Figure 21 lists some of the knowledge and skills required for professional competences in various IT architecture disciplines. We want to point out a number of observations of this diagram.

First, the diagram only shows operational knowledge and skills, by which we mean knowledge and skills required by IT architects designing particular business-IT systems. We extend this with strategic knowledge and skills listed later, in a separate diagram.

Second, different companies have different variations of this list, so Figure 21 should only be viewed as illustrative. At the same time, many topics in the diagram are familiar to students of computer science (the lower part of the figure) and students of business informatics (the upper part of the figure), and so the diagram is not very surprising either. Nevertheless, the diagram allows a number of interesting observations to be made:

– At the business level most of our sources mention the need for knowledge and skills for business process modelling and design, which covers the behavioural aspect of the organisation. They also mention the need for knowledge and skills in the design of administrative organisation, which covers the data and communication aspects at the business level. Internal control covers one quality aspect at the business level, namely security, or more specifically the integrity and correctness of the processes, data flow and data storage at the business level. Except internal control, all knowledge and skills at the business level concerns design, and there is a separate column of knowledge and

skills about the design process itself, such as notations, tools and techniques to perform the organisational design tasks.

– At the enterprise software level there is a rather unremarkable list of topics ranging from programming to technical design. Quality aspects other than security are not mentioned as separate topics by our sources. In addition, there is a column of skills and knowledge concerning the software design process itself, including notations, tools, techniques and frameworks to do software design.
– At the infrastructure level we see a large number of infrastructure components, such as operating systems and server technology. Where at the enterprise software and business levels we saw design knowledge and skills, here we see product knowledge and skills.

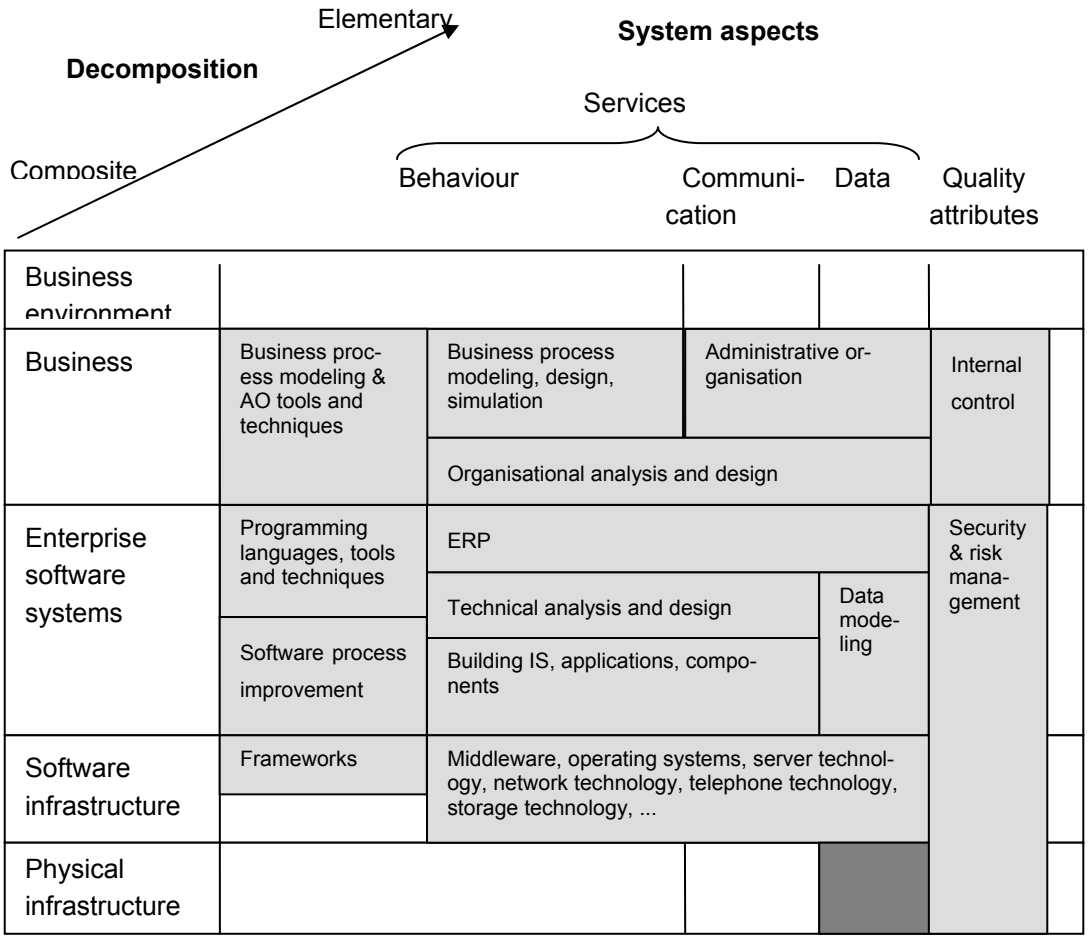


Figure 21 A sample of operational professional knowledge and skills for IT architects

Third, the design knowledge and skills map quite well to the GRAAL framework. They neatly span the three functional aspects of the GRAAL aspects (behaviour, communication and data) but ignore most quality aspects except security; and they also span the decomposition dimension, as it is about how to decompose a business or software system into components. Along the phase dimension of GRAAL (Figure 10) design knowledge and skills are located at the conception and building phases.

Moving to the infrastructure level we see mostly product knowledge rather than design knowledge. Product knowledge does not restrict itself to one aspect, is less concerned with how a product is decomposed and in the phases of a system focuses on the buying phase. So at this level, the GRAAL framework is less relevant.

Fourth, there are a number of blank spaces in Figure 21, which become apparent if we summarize the knowledge and skills in their major categories, in Figure 22. Here, design knowledge and skills includes knowledge of the design objects (the organisation or software) as well as of the tools and techniques for designing these objects. Filling in the blank spaces, we get Figure 23.

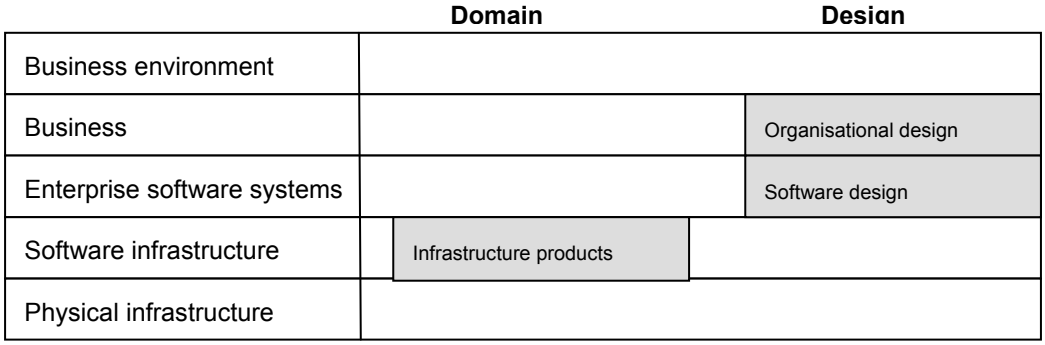


Figure 22 Major categories of operational knowledge and skills: Blank spaces

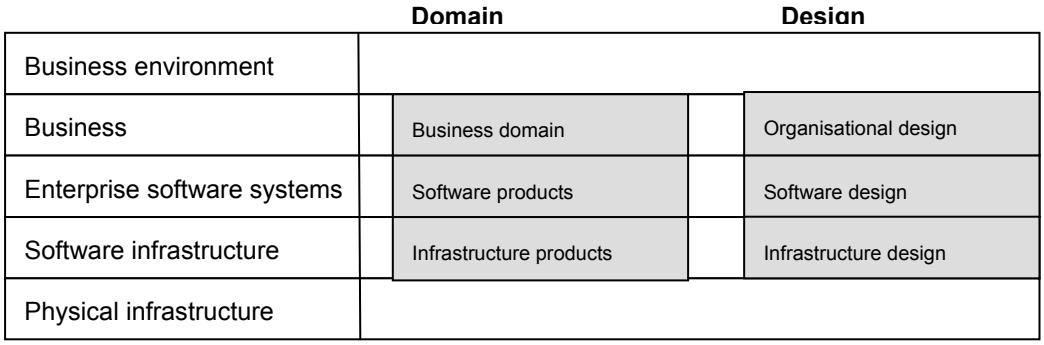


Figure 23 Major categories of operational IT architect knowledge and skills

At the infrastructure level, Figure 23 shows that design knowledge and skills are relevant but concern integration of infrastructure products bought on a market. At the enterprise software level, a lot of software is still custom made (in-house or outsourced) but ready-made products are playing an ever larger role. At the business level, knowledge and skills pertaining to the particular business domain are as important as business design knowledge and skills.

This concludes our remarks about operational IT architect knowledge and skills. Expanding now to include strategic knowledge and skills too, we have found that our sources mention various high-level skills:

– The ability to formulate a strategic vision of IT
– The ability to mutually align IT strategy and business strategy.
– The ability to identify opportunities that IT offers for the business

– The ability to align IT to business goals

We summarize this as strategic alignment knowledge and skills and complete our diagram of IT architect knowledge and skills as shown in Figure 24.

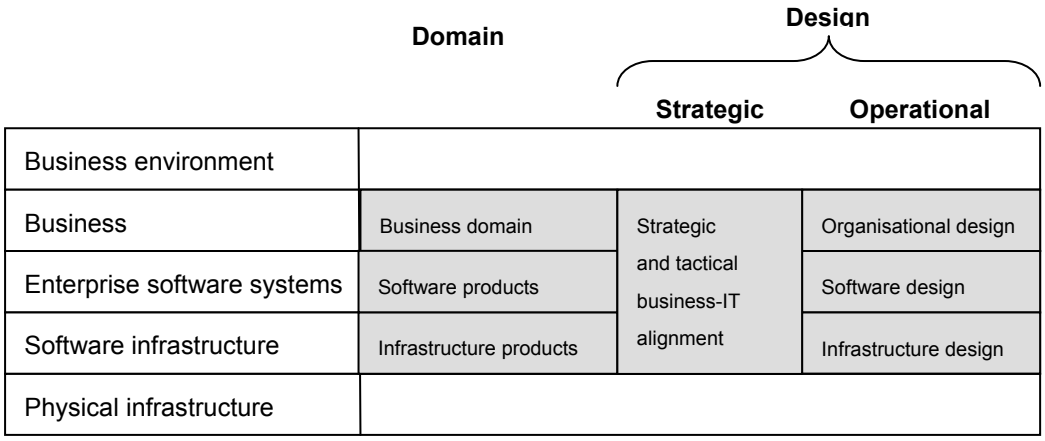


Figure 24 Strategic and operational IT architect knowledge and skills

This sums up the two major classes of IT architect knowledge, namely domain knowledge and skills and design knowledge and skills. These can be related quite easily to the architecture disciplines discussed in Chapter 4 (Figure 25).

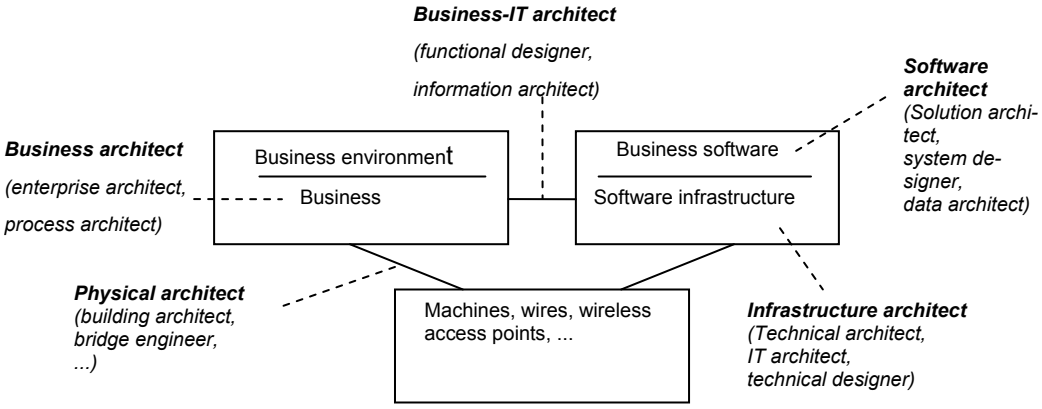


Figure 25 Architecture disciplines (from Figure 19, Chapter 4)

– Domain knowledge and skills
  – In the **business layer** this concerns the specific business the architect works in. This may be manufacturing domain, a government domain, or a financial domain such as banking or insurance. This is relevant for business architects and business-IT architects, as well as their variants such as enterprise architects, process architects, functional designers and information architects. The importance of domain knowledge at this level explains why these architects may come "from the business" and have received additional schooling in IT, or come from IT but have received additional schooling in business domain knowledge.

  - In the **enterprise software layer,** domain knowledge and skills are software product knowledge and skills in how to install and integrate these products with other enterprise software. This includes for example knowledge of ERP systems and of other commercial off-the-shelf software packages useful for the business. This is relevant for software architects, including software designers and data architects.
  - In the **infrastructure layer,** domain knowledge and skills includes knowledge of infrastructure components such as operating systems, middleware, server technology network software etc. and skills in installing and integrating these components in one infrastructure. This knowledge is relevant for infrastructure architects.
- Design knowledge and skills
  - At the **operational level** architects in any domain should have knowledge and skills in the relevant design techniques: Process design techniques for process architects, administrative organisation design techniques for business designers, information modelling and analysis techniques for information architects, etc. In addition, architects should be familiar with tools and processes to be used in architecture design in their domain.
  - At the **strategic level** we find enterprise architects who should be able to analyze a business strategy and compose an IT strategy, who should be able to make enterprise-wide models of services and information flows, etc.

The above discussion is organized according to the layers of the GRAAL framework. As pointed out in Section 4.1, security is an aspect that cuts across all layers, and hence security architects have not been mentioned in the above discussion. But for the security domain too we can identify the same kinds of architectural knowledge and skills: Products, operational design techniques, strategy.

Due to the prominence of design competences for architects, some of our sources mention some general design knowledge and skills such as

- solution design
- requirements engineering
- requirements management
- knowledge of architecture principles
- sourcing
- cost/benefit analysis of solutions

These competences are not particularly related to architecture and could also be viewed as general professional competences, treated next.

6.2 General professional competences

IT architects must work with many different stakeholders and, as all professionals, should have more general competences to do this. Our sources mention competences in the areas of management and consultancy. Most frequently mentioned are communication skills and leadership skills. This can be explained by the fact that many architects must communicate with many stakeholders to acquire information about current architecture, and this requires excellent communication skills. Moreover, many architects must also con-

vince many stakeholders of the value of architecture decisions and this requires leadership skills.

The complete list of general professional competences mentioned by our sources for architects follows.

- Management skills
  - Oral communication
  - Written communication
  - Leadership
  - Project management
  - Program management
  - Portfolio management
  - Change management
- Consultancy skills
  - Negotiation skills
  - Didactic skills
  - Coaching skills

What skills a competent architect actually needs depends on his or her role in the organisation. Is she allocated to a project, a program, to a business unit, to a central staff department? The organisational embedding of architects is outside the scope of this document and we will not elaborate on this further.

6.3 Cultural characteristics

Cultural characteristics tell us “how we do things around here”. They define a group identity and they are learned by slow enculturation in a group. They contain norms, values and background knowledge shared by all members of the group. The list of cultural characteristics of IT architects mentioned by our sources is short but highly significant:

- Business orientation
- Customer orientation

This means that someone cannot become an architect if she is not able to understand the problems and goals of the business or of its customers, and is not able to subscribe to the goals of the business or its customers. This is particularly important for IT architects with an IT background; but for but for people with a background in the business it is important too, because it means that IT architecture is not an escape route for people who discover they like IT more than the business. These two cultural characteristics determine the attitude by which an IT architect goes about her work and this is a crucial element in IT architecture competence.

6.4 Personality characteristics

The least changeable elements of professional competence are the personality characteristics of the professional. These determine to a large part the attitude of the professional,

which in turn is an important determinant of professional competence. To classify the personality characteristics indicated by our sources, we use the five clusters of personality characteristics identified by psychologists (Goldberg 1990). Psychologists have identified these characteristics by clustering words that people use to characterize personalities. We classify the personality characteristics mentioned by our sources under these five headings:

- Extraversion
  - Communicative
  - Initiative, energy
  - Willing to take on challenges
  - Persuasiveness
- Agreeableness
  - Team player
  - Empathic
  - Able to listen
  - Trustworthy
- Dependability, Conscientiousness
  - Analytic
  - Organized and systematic
  - Decisive
  - Result-oriented
  - Reliable
  - Accurate
  - Perseverance
- Emotional stability
  - Independent
  - Stress-resistant
- Intellect
  - Creative
  - Able to abstract
  - Ability to learn from experience

The personality traits most frequently mentioned are communicativeness, team player, analytic, creative and able to abstract.

What is one to make of this? IT architects must be a rare breed, supermen and –women that would do well in any profession. We can identify two dimensions in the above list.

- Masculine-feminine
  - The masculine architect is result-oriented, decisive, and persuasive.
  - The feminine architect is a team player, empathizes with other people and listens well.
- Visionary – analytic
  - The visionary architect is a dreamer who invents creative solutions at a high abstraction level

– The analytic architect elaborates architectures in an organized, accurate and systematic way.

All of them are, of course, independent and stress-resistant and live a life of continuous learning by reflecting on their own activity. But it is not possible to be a masculine and feminine architect at the same time, or to be a visionary and analytic architect at the same time. Combinations of the two dimensions are possible: A visionary masculine or feminine architect, or an analytic masculine or feminine architect.

7 Summary and recommendations

Our survey has shown that beneath the diversity of terminology and frameworks for IT architects there is a convergence of views. IT architects can have different specializations, called disciplines in this book, which focus on the business, on software or on their mutual alignment. Examples of disciplines with a business orientation are enterprise architects and process architects; examples of disciplines with a software orientation are software architects, system designers and data architects; and example of disciplines that focus on the mutual alignment of business and software are information architect and functional designer.

Professional knowledge and skills of these different architect disciplines may focus on the business domain, on software products, or on strategic or operational design of business and IT solutions, as described in chapter 6. In addition to these specialized professional competences, an IT architect must have general professional skills such as management and consultancy skills.

Very important too, but harder to change, are cultural competencies such as business orientation and customer orientation, and personality characteristics such analytical competence, creativity, the ability to abstract from details, the ability to learn and the ability to communicate.

One area not discussed in this book is that of job roles for IT architects. For example, IT architects may play the role of project architect, programme architect or enterprise architect (now used as a name for the person responsible for IT architecture at the enterprise level). These roles require specific competences such as change management or portfolio management. Job roles for IT architects have not been considered in this book because this is a general topic bordering on management science and organisation design, about which so far, not enough is known to make general claims about recurring and stable patterns in different businesses.

How can the underlying convergence of views on IT architect competencies identified in this book be used? We do not intend to define a standard to be followed by others; rather, we have presented a conceptual framework and terminology that allows companies to analyze their own IT architect disciplines and companies and position it with respect to the disciplines and terminology of others. This is useful for client companies trying to interpret the differing and sometimes confusing variety of disciplines and terms used by consultancy companies. This book helps client companies understand what consultancy companies offer.

Furthermore, this book also helps consultancy companies and others to position their own IT architecture disciplines and terminology to those of reference models like the NGI and TOGAF frameworks. For example, we have shown that the GRAAL framework is a greatest common denominator of the important well-known other architecture frameworks. By acting as a central reference point, GRAAL can be used by any company to position its own architecture framework with respect to that of others.

8 Bibliography

1. G.J. Bergenhenegouwen, E.A.M. Mooijman, H.H. Tillema. *Strategisch opleiden en leren in organisaties*. Kluwer, 2de druk, 1999. (In Dutch)

2. B. S. Blanchard and W. J. Fabrycky. *Systems Engineering and Analysis*. Prentice-Hall, 1990.

3. J. op de Coul. *Taken, functies, rollen en competenties in de informatica*. Den Haag: Ten Hagen Stam, 2001. (in Dutch).

4. G. ten Dam, J. Vermunt. "De leerling". In N. Verloop en J. Lowyck (red.), *Onderwijskunde*. Wolters-Noordhoff, 2003. Pages 151-194. (In Dutch)

5. P. van Eck, H. Blanken, R. Wieringa. "Project GRAAL: Towards Operational Architecture Alignment" *International Journal of Cooperative Information Systems*. Vol. 13 No. 3, Pages 235-255. September 2004.

6. GRAAL. The GRAAL project white papers, http://graal.ewi.utwente.nl/whitepaper.php.

7. A.D. Hall. *A Methodology for Systems Engineering*. Van Nostrand, 1962.

8. A.D. Hall. "Three-dimensional morphology of systems engineering". *IEEE Transactions on System Science and Cybernetics,* SSC Vol. 5 No. 2, Pages 156–160, 1969.

9. D. Harel, A. Pnueli. "On the development of reactive systems". In K. Apt (ed.), *Logics and Models of Concurrent Systems,* Springer, 1985. Pages 477-498.

10. J. Henderson and N. Venkatraman. "Strategic alignment: leveraging information technology for transforming operations". *IBM Systems Journal,* Vol. 32 No. 1, Pages 4-16, 1993.

11. R.L. Goldberg. "An Alternative "Description of Personality": The Big-Five Factor Structure". *Journal of Personality and Social Psychologs*. Vol. 59 No. 6, Pages 1216-1229, 1990.

12. B. Iyer and R. Gottlieb: "The four-domain architecture: an approach to support enterprise architecture design". *IBM Systems Journal*. Vol. 43 No. 3, Pages 587-597, 2004.

13. H. Jonkers, M.M. Lankhorst, R. van Buuren, S. Hoppenbrouwers, M. Bonsangue, L. van der Torre, Concepts for Modelling Enterprise Architectures, *International Journal of Cooperative Information Systems,* Vol. 13 No. 3, Pages 257–288, 2004.

14. P. Kruchten. "The 4+1 view model of architecture". *IEEE Software*, Vol. 12 No. 6, Pages 42–50, November 1995.

15. R. Maes. *Reconsidering information management through a generic framework*. PrimaVera Working Paper 99-15, University of Amsterdam, Department of Accountancy and Information Management, 1999.

16. R. Maes, D. Rijsenbrij, O. Truijens and H. Goedvolk: *Redefining Business – IT Alignment Through a Unified Framework*, PrimaVera Working Paper 2000-19, University of Amsterdam, Department of Accountancy and Information Management, 2000.

17. J. Sowa and J. Zachman: "Extending and formalizing the framework for information systems architecture" *IBM Systems Journal,* Vol. 31 No. 3, Pages 590–616, 1992.

18. C. Steghuis, K. Voermans, and R. Wieringa. "Interview report on the competences of the ICT-Architect". NAF Report, 2005.

19. TOGAF Enterprise Edition, version 8.
http://www.opengroup.org/architecture/togaf8-doc/arch/

20. K. Voermans, C. Steghuis and R. Wieringa (2005). Competences of the ICT architect. Dutch Architecture forum.

21. R.J. Wieringa. ``Postmodern software design with NYAM: Not yet another method", In M. Broy and B. Rumpe, editors, *Requirements Targeting Software and Systems Engineering*, pages 69–94. Springer, 1998. Lecture Notes in Computer Science 1526.

22. R.J. Wieringa, H.M. Blanken, M.M. Fokkinga, and P.W.P.J. Grefen. ``Aligning application architecture to the business context." In *Conference on Advanced Information System Engineering (CAiSE 03)*, pages 209–225. Springer, 2003. LNCS 2681.

23. R.J. Wieringa and P.A.T. van Eck and D. Krukkert (2005) "Architecture Alignment". In: *Enterprise Architecture at Work. Modelling, Communication and Analysis*. Springer. Chapter 9, 2005.

24. J.A. Zachman. "A framework for information systems architecture" *IBM Systems Journal*, Vol. 26 No. 3, Pages 276–292, 1987.

9 Appendix

This appendix contains two tables (in Dutch) from the NGI report (Op de Coul 2001). The first table lists all tasks for the four architecture disciplines discussed in Section 2.2. The second table lists all professional competences that are mentioned in the description of the Information Architect discipline in the NGI report.

Informatie-architect	Data-architect	Software architect	Architect tech. infrastr.
Taakgroep: Kaderstelling, beleid en architectuur			
Opstellen informatieplan			
Vaststellen informatie-architectuur	Vaststellen informatie-architectuur		
Opstellen plan voor interne controle			Opstellen plan voor interne controle
Opstellen bev-eiligingsplan			Opstellen bev-eiligingsplan
			Bepalen architectuur van de technische infra-structuur
			Bepalen standaarden technische infrastruc-tuur
			Bepalen normen ge-bruik en beheer tech-nische infrastructuur
Taakgroep: Ontwikkelen, ontwerpen en bouwen			
Probleem-oriëntatie informatievoorziening			
Onderzoeken organisa-tie-aspecten			
Analyseren verander-ingsvermogen			
Vaststellen gewenste situatie informati-evoorziening	Vaststellen gewenste situatie informati-evoorziening		
Opstellen objectmodel	Opstellen objectmodel		
Opstellen bedrijfsproc-essen model	Opstellen bedrijfsproc-essen model		
Opstellen infor-matieprocessen model			
Vaststellen informatie-systeem-architectuur		Vaststellen informatie-systeem-architectuur	
Opstellen gegevens-model	Opstellen gegevens-model	Opstellen gegevens-model	
		Ontwerpen functionele specificaties	
		Ontwerpen applicati-estructuur	
		Ontwerpen program-maspecificaties	
			Ontwerpen netwerk
			Ontwerpen configuratie van computersystemen, servers
			Selecteren hardware
			Selecteren systeem-software
			Selecteren beheersys-temen

Informatie-architect	Data-architect	Software architect	Architect tech. infrastr.
			Ontwikkelen beheersystemen
			Opstellen uitwijkplan
Taakgroep: Invoeren, implementeren			
		Maken testplan	Maken testplan
		Uitvoeren test	
			Bepalen invoeringsstrategie
			Bepalen migratiestap
			Bepalen conversiestrategie
Taakgroep: Beheren, exploiteren, onderhouden			
	Beheren metagegevens		
	Beheren configuratie-items		
		Beheren functionaliteit applicatie	
		Bepalen onderhoud op applicaties	
			Beheren prestatiekenmerken netwerk
			Beheren prestatiekenmerken server
			Beheren systemen voor gegevensopslag
			Structureel oplossen systeemstoringen
Taakgroep: Algemeen van toepassing op K, O, I en B,			
	Opstellen normen, criteria, randvoorwaarden		
	Opstellen kwaliteitsplan		

Table 11 Tasks of the four architecture disciplines discussed in Section 2.2.

Vaktechnische competentie	**Engelse benaming in Figuur 3**	**Toelichting**
Bedrijfskunde	Management science	Algemene aspecten van het functioneren van organisaties in een bedrijfskundig perspectief.
Organisatieleer	Organisation science	Algemene aspecten van organisaties.
Administratieve organisatie	Administrative organisation	Algemene aspecten van de structuur en de inrichting van organisaties gelet op het verkrijgen van adequaat beheerde en beheersbare bedrijfsprocessen.
Methoden en technieken voor interne controle, zowel wat betreft organisatorische als technische	Methods and techniques for internal control	Methoden en technieken gericht op het verkrijgen van adequaat beheerde, beheersbare en controleerbare bedrijfsprocessen.
Methoden en technieken voor beveiliging, zowel wat betreft organisatorische als technische	Methods and techniques for security	Methoden en technieken gericht op het voorkomen van toegang tot gegevens door niet-geautoriseerde personen, alsmede het voorkomen van het verlies van gegevens.
Organisatiegerichte analysemethoden en –technieken	Methods and techniques for analyzing organisations	Methoden en technieken voor het analyseren van organisaties en bedrijfsprocessen.
Organisatiegerichte ontwerpmethoden en –technieken	Methods and techniques for designing organisations	Methoden en technieken voor het ontwerpen van organisaties en bedrijfsprocessen.
Mogelijkheden van de informatietechnologie	Possibilities of IT	De algemene mogelijkheden van de informatietechnologie ter ondersteuning of de uitvoering van bedrijfsprocessen.
Architectuurprincipes	Architecture principles	Algemene principes voor het structureren van objecten.
Technisch gerichte analysemethoden en –technieken	IT-related methods and techniques for analysis	Methoden en technieken voor het analyseren van technische processen in geautomatiseerde systemen.
Technisch gerichte ontwerpmethoden en –technieken	IT-related methods and techniques for design	Methoden en technieken voor het ontwerpen van technische processen in geautomatiseerde systemen.
Methoden en technieken voor applicatie(component)-bouw	Methods and techniques for building application (components)	Methoden en technieken voor het bouwen (realiseren) van componenten van informatiesystemen.
Gegevensmodellering	Data modeling	Methoden en technieken voor het defini'ren en structureren

Vaktechnische competentie	Engelse benaming in Figuur 3	Toelichting
		van gegevensverzamelingen.
Gegevens-analyse	Data analysis	Methoden en technieken voor het onderkennen en defini'ren van gegevensverzamelingen.
Beheer van informatiesystemen	Management of information systems	Algemene principes van het blijvend laten voldoen van informatiesystemen aan de eisen en wensen van gebruikers of beheerders van (andere) componenten van de informatievoorziening, inclusief de daaraan gestelde technische eisen.
Beheer van netwerken	Network management	Algemene principes van het blijvend laten voldoen aan de eisen en wensen van gebruikers of beheerders van (andere) componenten van de informatievoorziening.
Beheer van servers	Server management	Algemene principes van het blijvend laten voldoen aan de eisen en wensten van gebruikers of beheerders van (andere) componenten van de informatievoorziening.
Realiseren van organisatie wijzigingen	Realising organisational changes	Algemene principes van het veranderen (wijzigen) van structuren, werkwijzen, etc. in een (gebruikers)organisatie.
Opstellen van gebruikers documentatie	Writing user documentation	Algemene principes van het schrijven van documentatie in 'gebruikerstermen'.
Kwaliteitsmanagement	Quality management	Algemene principes van kwaliteitszorg en het managen van de processen om kwalitatieve en gewenste oplossingen te verkrijgen

Table 12 Architecture competences of the Information Architect (adapted from Op de Coul 2001)