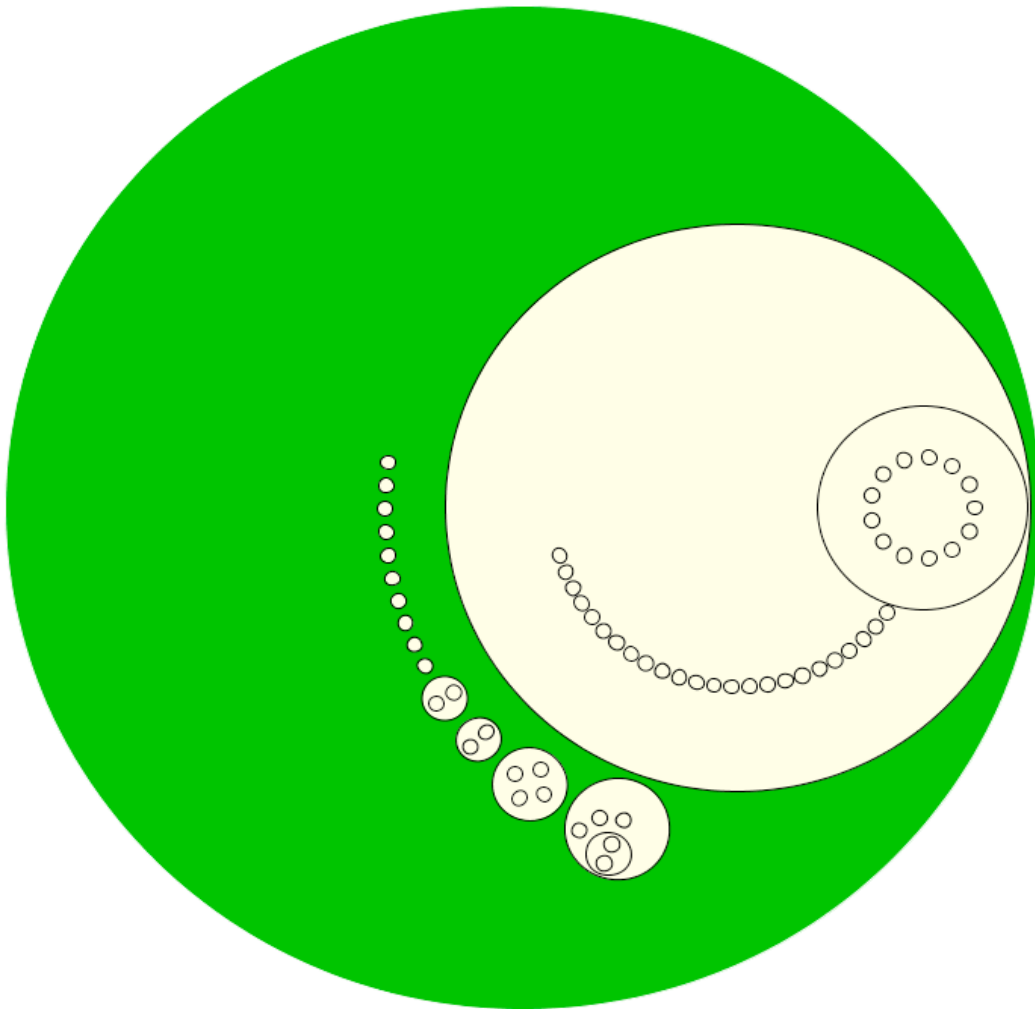


Enterprise Agility

An Integrated Approach



Author: Lars Hansen

E-mail: lars.hansen@gmail.com

Supervisor: John Gøtze

Master Thesis at IT-Universitetet

November 2008

The picture on the front-page is a graphical illustration of TOGAF 8 as an ontology.

The picture is generated with the “Fly The Mothership” function in the Semantic Web Ontology Editor (Swoop) application.

Foreword

This publication is a thesis submitted to the IT-University for a Master of Science in Information Technology, Ebuss. The thesis is supervised by John Gøtze.

First and foremost, I would like to thank all the people who has somehow contributed to this thesis. In particular, I would like to thank my supervisor John Gøtze. Since I followed his course in Enterprise Architecture during my second term on the IT-University, I was certain that my master thesis would be written within this particular field. In this, I have no regrets. So I would like to thank him for his enthusiasm, his patience and most importantly for pushing me to do my best.

Moreover, I would like to thank the people who have been interviewees for this thesis, despite their busy work schedules. The input from specialists within the field has been invaluable.

Finally, it is my hope that the reader will enjoy this thesis.

Lars Hansen

IT-Universitetet, November 2008

Abstract

This thesis concludes my Master of Science, Information Technology in Ebuss. It has been the basic tenet for this thesis, that enterprises are operating in world that is becoming ever more fast-paced and unpredictable. This reality calls for a greater emphasis on the ability to respond effectively to changes, rather than on rigid long-term planning. As such, the ability to act in an agile fashion will be a critical competence for enterprises in navigating this new world. It was with these observations in mind, that I sought out to establish an integrated approach to EA, SOA and BPM, which can be leveraged for agility. The focal point was to enable enterprises to better handle business changes, driving changes in business processes and their underlying information systems.

Keywords: *enterprise, agility, business process management, enterprise architecture, service-oriented architecture, alignment, governance, meta-data management, ontology, meta-model, rdf, rdfs*

The thesis was done through a theoretical literature study, supplemented by qualitative interviews with people considered experts within relevant domains. Moreover, a simple enterprise meta-model was devised and implemented in an ontology.

The main findings of this thesis are:

- The relationship between BPM and SOA is often sold on the idea of a business analyst being able to analyse, design and deploy business processes with little or no intervention from IT. I did however find significant semantic gaps in the chain from analysis to deployment. The primary advantage of using BPM and SOA together is to decompose business processes from their underlying implementation details, rather than to automate software development
- Stake-holders, such as business analysts, system architects and system developers still needs to come together in order to bridge the semantic gap between business and IT. A process for service design was devised, including using BPM for domain decomposition.
- Coupling remains an issue, even in a loosely coupled service-oriented architectures. I especially pointed towards the problem of semantic coupling. Managing residual coupling calls for sound meta-data management and governance.

- BPM and SOA together requires long-term planning, commitment and management far beyond the individual BPM-SOA project. I found EA to be an excellent tool for establishing BPM and SOA. The focus on alignment is however at the cost of agility. I devised some modifications to the TOGAF ADM that would make it possible to better balance agility and alignment.
- Integrating EA, SOA and BPM for agility requires visibility across all EA artefacts. An architecture based on BPM-SOA contains a rather large amount of components, each having their own life-cycle to be managed. I found that a shared neutral language connecting EA artefacts was needed. This language could bridge the many small worlds of domain specific models into a consolidated model of the enterprise.
- A case scenario was described in order to codify important findings in the thesis
- Finally, I demonstrated initial feasibility of using ontologies and ontology representation languages to establish the common language for EA. The ontology was based on RDFS.

The overall conclusion of this thesis is, that I see meta-data management and meta-data integration through the creation of a common architecture language, as absolutely vital in integrating BPM, SOA and EA around the theme of agility.

Table of Contents

Thesis Problem and Methodology.....	1
1.1 Motivation.....	1
1.2 Thesis problem.....	4
1.3 Delimitation.....	5
1.4 Clarification of Problem Definition.....	6
1.5 Methodology.....	7
1.6 Thesis structure	10
Module 1 – Background and Concepts.....	12
Chapter 2 – Enterprise Agility.....	13
2.1 Definition of Enterprise.....	13
2.2 Definition of Enterprise Agility.....	14
2.3 Enterprise Agility and Information Technology.....	17
2.4 Agility versus Alignment.....	19
2.5 Enterprise Agility and Agile Software Development.....	21
2.6 Chapter Summary.....	23
Chapter 3 – Understanding the parts.....	24
3.1 Enterprise Architecture	24
3.2 Business Process Management.....	35
3.3 Service-oriented Architecture.....	43
3.4 Chapter Summary.....	53
Module 2 – The Sum of the Parts.....	55
Chapter 4 – BPM and SOA.....	56
4.1 The Vision of BPM-SOA.....	59
4.2 The Model-driven View.....	62
4.3 The Methodology-driven View	81
4.4 Is BPM the Business Case for SOA?.....	95
4.5 Chapter Summary.....	97
Chapter 5 – Contribution of Enterprise architecture	99
5.1 Architecture Development in the Context of BPM-SOA.....	100
5.2 Enterprise Architecture and Agility.....	102
5.3 Building a Foundation for Execution.....	113
5.4 The Integrated Approach	123
5.5 Challenges of Integrated Enterprise Modelling.....	126
5.6 Chapter Summary.....	135
Module 3 – Connecting the Dots.....	137
Chapter 6 – Implementing a Common Language.....	138
6.1 Ontologies.....	139
6.2 Ontology Representation.....	142
6.3 Using RDFS to Implement the Meta-model.....	149
6.4 Chapter Summary.....	151
Chapter 7 – The Project Scenario.....	152
7.1 Project Scenario – T.A.X.....	152
7.2 The Common Language.....	156
7.3 Personal Reflections.....	167
Module 4 – Conclusion, Perspectives and Criticism.....	170
Chapter 8 – Conclusion, Perspectives and Criticism.....	171
8.1 Conclusion	171
8.2 Perspectives.....	175
8.3 Criticism.....	178

List of Appendices

A - Notice on Changed Problem Statement.....	183
B - Indicators of BPM-SOA Interest.....	184
C - Thesis Interviews.....	186
D - The Zachman Framework	187
E - The TOGAF Architecture Development Method	188
F - Note on Model-driven Architecture (MOF & XMI).....	189
G - The Enterprise Meta-model.....	194
H - Meta-model Concepts (Class hierarchy).....	195
I - Meta-model Relations (Object properties).....	200
J - The RDF-Gravity Legend Map.....	201
K - Bibliography.....	202

Figures

Figure 1: Strategic Alignment Model [Adapted from Venkatraman & Henderson:479].....	19
Figure 2: The Zachman Framework [ZIFA].....	29
Figure 3: Architecture Development Method [TOGAF:20].....	33
Figure 4: Process Life-Cycle [SAP].....	37
Figure 5: 7FE BPM Framework [Adapted from Jeston & Nellis, 2008, 1:63].....	41
Figure 6: Conceptual Service Model [Own production].....	46
Figure 7: Service Interaction Model [Arsanjani].....	48
Figure 8: Abstraction of Business Processes [Own production].....	52
Figure 9: BPM-SOA Stack [Kamoun].....	59
Figure 10: Modelling Value-chain [Own production].....	66
Figure 11: Language Description Elements [Own production].....	71
Figure 12: Business versus Service Modelling.....	78
Figure 13: Domain Decomposition [Own production].....	85
Figure 14: Simple Business Process [Own production].....	87
Figure 15: Service Specification Requirements [Own production].....	89
Figure 16: Semantic Service contract [Own production].....	92
Figure 17: Alignment Model for the ADM [Own production].....	103
Figure 18: Decomposing Business and IT alignment [Sousa, Pereira & Marques:35]....	105
Figure 19: TOGAF ADM with Phase Steps [TOGAF:21].....	107
Figure 20: Architecture Change [Own production].....	112
Figure 21: Operating Models [Adapted from Weill, Ross & Robertson:29].....	115
Figure 22: The Effect of Operating Models on BPM-SOA [Own production].....	117
Figure 23: Foundation Architecture – Aligning Business and IT [Doucet et. al:5].....	120
Figure 24: Extended Architecture [Doucet et al.:6].....	121
Figure 25: Modified Architecture Development Method [Own production].....	123
Figure 26: Meta-model [Own production].....	132
Figure 27: Ontology Layers [Own production].....	141
Figure 28: Semantic Web Layers [Passin:14].....	143
Figure 29: Example of RDF Triple [Own production].....	144
Figure 30: Complex RDF Example [Own production].....	145
Figure 31: Example Business Process [Own production].....	159
Figure 32: Simple Process Overview [Own production].....	160
Figure 33: Line-of-Sight - Architecture Domains [Own production].....	161
Figure 34: Line-of-sight - Strategy Domain [Own production].....	163
Figure 35: Business Process Versioning [Own production].....	165
Figure 36: Architecture Versioning [Own production].....	166
Figure 37: UML Model Exchange [Lundell et al.:626].....	192
Figure 38: UML Model Exchange XMI 2.0+ [Lundell et al.:628].....	193
Figure 39: Strategy Class Hierarchy [Own production].....	195
Figure 40: Business Architecture Class Hierarchy [Own production].....	196
Figure 41: Application Architecture Class Hierarchy.....	197
Figure 42: Data Architecture Class Hierarchy [Own production].....	198

Tables

Index of Tables

Table 1: Enterprise Change Process [Own production].....	16
Table 2: BPM Initiation Drivers - [Own production].....	42
Table 3: Tight versus Loosely Coupled Systems [Kaye:133].....	50
Table 4: BPM versus SOA [Own production].....	56
Table 5: Process Languages Feature Comparison [Own production].....	76
Table 6: Life-cycle Phases of Services in SOA [Own production].....	81
Table 7: Benefits of SOA [Own production].....	96

Thesis Problem and Methodology

1.1 Motivation

The basic premise for this thesis is that enterprises of today are operating in an increasingly complex environment, characterised by constant change. It is sometimes said that *“change is the new constant”*. But not only has the pace quickened; enterprises are also facing increased competitive pressure. In a globalized world, with increasingly demanding marketplaces, enterprises are faced with the daunting task of creating the capability to respond to changes, and to continuously improve business. This issue is highlighted in a recent Garner EXP survey: [Gartner, 2008]

“Eighty-five percent of chief information officers (CIOs) see significant change coming over the next three years as they look to meet rising business expectations for IT to make the difference in their enterprise strategy,”

The perhaps most important factor shaping the need for agility is globalisation. Globalisation itself, can be considered an umbrella term, covering a lot of different tendencies. Globalisation is a subject causing much discussion and controversy, but in this context globalisation will be seen as *“a tendency towards a higher degree of co-operation integration and dependence across national state boundaries”*.¹ [OEM:5] Globalisation can especially be seen expressed in advances in transportation-, communication- and information technologies. But not only technological factors are at play. Deregulation has had its effect too. Barriers once erected between countries or regions, as well as barriers between public and private sector enterprises are being broken down. So enterprises who once were protected by legal or trade barriers are now subject to fierce competition in an open market.

¹ Translated from danish to english. Original quote is: *“Globalisering er udtryk for en større grad af samarbejde, integration og afhængighed på tværs af landegrænserne.”*

Globalisation does not only increase competition, but also makes the environment more unstable and unpredictable. This development towards the more unstable and unpredictable, can be described more formally through some of Dave Snowden's work. Snowden is engaged in describing how to make sense, and make decisions in different types of systems. In his "*Complex acts of knowing*" he differentiates between complicated and complex systems.² [Snowden:7-8] Complicated systems are reducible and can be subject to traditional cause-and-effect analysis. Complex systems in contrast, involves many different agents (an agent denoting everything that has one or more identities). Complex systems are irreducible, and cannot be subject to traditional cause and effect analysis.

In it-self, complexity is nothing new. But as customers, competitors, suppliers, and partners all react to the new opportunities and threats arising from globalisation, the environment becomes more volatile. In order to position themselves, the various stake-holders all change positions vis-à-vis each other, by constantly re-configuring their value chains. Obvious examples are dis-intermediation, where online technologies allows for skipping intermediaries in the value chain, providing a more direct producer / end-user relationship. In other situations, intermediation takes place, where new intermediaries appear, such as travel-bookers, shopping portals etc. Increased use of sourcing and outsourcing also impacts the value chain. Moreover, many of the new competitors entering into ones home market, often have access to a very different set of resources and competencies, compared to the competitors originating in the geographical home market. This also carries the risk of inducing non-linear changes, in the form of disruptive innovations, i.e. innovations that drastically alters the rules on the market. This notion has been popularized by Thomas L. Friedman in his bestseller "*The World is Flat*". The central theme of his book is that the economic playing field has been leveled. Globalization he says, "*is shrinking the world from a size small to a small tiny and flattening the playing field at the same time*" [Friedman:10]

² Snowden further divides complicated systems into *known* and *knowable*. In known systems cause and effect has already been identified, and in knowable systems we can identify cause and effect by using enough resources for the analysis.

These developments creates a much more dynamic, and unpredictable environment. In such a complex environment, enterprises can no longer identify cause and effect, and will increasingly rely on managing patterns, rather than knowing exactly what to do next. Snowden observes: *"In the complex domain we need to identify the early signs of a pattern forming and disrupt those we find undesirable while stabilising those we want"*. [Snowden:8] In such an environment, greater value is placed on the ability to detect patterns and to respond fast, rather than on rigid long time planning. It appears that something needs to be done in order to adapt to this environment.

1.2 Thesis problem

Having realized the growing need for change, enterprises are increasingly embracing the concept of agility, and are employing a wide range of strategic initiatives to improve agility. Sometimes used as standalone initiatives, and sometimes in combined effort. Among these initiatives are:

- Enterprise Architecture (EA)
- Service Oriented Architecture (SOA)
- Business Process Management (BPM)

The central theme of this thesis is, that EA, SOA and BPM each concerns different aspects of agility in an enterprise. The relationship between between these initiatives are however not very well understood. The main argument of this thesis is that the three initiatives can be brought together to improve agility, but also that something is missing in order to integrate them. Thus, the overall goal of the thesis is to devise an integrated approach to EA, SOA, and BPM. ³

The research problem for this thesis is:

How can enterprises integrate EA, BPM and SOA for agility?

³ It should be noted that the thesis problem has been modified. See appendix A

1.3 Delimitation

There are of course several elements to enterprise agility, and “*IT i Praksis*”, by Rambøll Management defines four elements: [Rambøll;6]

- Structure
- People
- Processes
- Technology

To define an appropriate scope, for this thesis, I will however not deal with all four elements. The focal point of this thesis will be business changes, driving changes in business processes and their underlying information systems. Business processes are core elements of an enterprise, and include interactions between humans and systems.⁴ An enterprise can be thought of as “*an aggregation of processes and the resources that comprise those processes*” [Bloomberg & Schmelzer:61] Smith & Fingar goes as far as saying that: “*business processes are the business*”. [Smith & Fingar:17] While this statement can be considered an oversimplification, business processes are without doubt of central value to an enterprise, because business processes defines “*how people and other resources work*”. [Bloomberg & Schmelzer:61]

But business processes cannot stand alone, as they are just the organising logic for the work being done. I see a natural relationship between processes and information systems; processes depends on information systems, and information systems are often designed or customised to support business processes. Moreover, business processes are often embedded within information systems, effectively making the business at the mercy of IT. [Smith & Fingar:13]

Deciding to look mainly at process, and technology has some limitations. Agility in this thesis is the capability to continuously perform incremental improvements to business processes. This means, that this thesis will not concern the transformation of enterprises through giant steps, but rather through series of small incremental steps. One could say that this view on agility is more small-bang than big-bang.

4 Being Human-System, Human-Human, or System-System interactions

1.4 Clarification of Problem Definition

Having three main entities to study in the thesis, there will be the important question of how to relate them to each other. An obvious approach would be to analyse the entities as three pairs, i.e. EA, vs. SOA, EA, vs. BPM, and BPM vs. SOA.

I have however chosen a different approach. Instead, I will perform the analysis by first looking at BPM and SOA combined under the umbrella term BPM-SOA. Bearing in mind, that the focal point of this thesis is agility in relation to business processes and information systems, I find it natural to treat them in combination. BPM for the business and SOA for the IT. There are also several indicators pointing towards BPM and SOA already being seen as complementary entities. Among these indicators are increased customer interest in combining BPM and SOA, vendor consolidation within the field, and a generally growing awareness around the combination of BPM and SOA. In appendix B, some of these indicators have been detailed.

Following the BPM-SOA analysis, I will proceed to analyse the relationship between BPM-SOA and EA. I see EA playing a very different role in regards to agility compared to that of BPM and SOA; EA is taking the long-term enterprise-wide look at resource utilisation in the enterprise. In some ways, this long-term view is an anti-thesis to agility, but I see huge synergies in using EA in combination with BPM and SOA. The integrated approach, combines BPM-SOA with EA, will be codified in a model.

But I will however also find, that something is missing from the equation. To be able to integrate EA, BPM and SOA there need be a shared language to understand the architecture as a whole. Thus, I will have to devise a way to define such a language. Moreover, the language should be possible to implement by electronic means. I will create a proof-of-concept implementation of such a language.

Finally, through a case scenario, I will codify the most important finding of this thesis, as well as demonstrate the use of the language implementation.

1.5 Methodology

1.5.1 Research Design

The overall purpose of this thesis is to create a clearer conceptual understanding of the relationship between enterprise agility, and an integrated approach to EA, SOA, and BPM. The research design, provides guidance about how to achieve this goal. It is my intent to identify potential problems in leveraging EA, SOA and BPM for agility, as well as pointing out solutions to said problems. I have therefore chosen to opt for combined approach to the research, by including two different research methods:

- **Explorative:** As the name implies, the explorative research design is about exploring new or unknown fields or phenomena. A specific branch of the explorative research methods concerns problem identification. Another important characteristic of the explorative research design is that it allows me to formulate work hypotheses. [Andersen:23-24] I will utilise these characteristics in combination; by formulating working hypotheses, which can be either validated or rejected, I will uncover possible problems in regards to leveraging EA, SOA, and BPM for agility.
- **Normative:** The normative research design is concerned with formulating solutions or ways to remedy identified problems. [Andersen:28] Based on the problems identified in my exploration of the problem domain, I will devise solutions to these problems. As I have decided not to include a case, the solutions will be of a more general nature, rather than with the purpose of intervening in a specific case or enterprise.

1.5.2 Hermeneutics

It is a hallmark of good research, that it can be considered objective. Objectivity is often contrasted to that of subjectivity (or bias versus non-bias). [Kvale:71] Unfortunately, the term *objective* is in itself a rather subjective term, as there are many different interpretations [Kvale:72]. A goal of this thesis is to perform research that has *freedom from bias*, i.e. is free from my personal prejudice, opinions, and perceptions as possible.

This would among other things require, that I - as a researcher - should be able to interpret the literature in an impartial, and unbiased way. This is however an attempted ideal. Hermeneutics, which is a field that is concerned with the interpretation of literature, highlights this point: "*Hermeneutics is an approach to the analysis of texts that stresses how prior understandings and prejudices shape the interpretative process*". [Denzin & Lincoln:27] As a researcher, I already possess a body of knowledge before writing this thesis, and this knowledge does not only affect how I look at the subject matter, but it will also impact how I interpret the literature that I encounter.

The problem of interpretative bias is especially relevant in relation to the subject that I am about to study. I consider enterprises *complex systems* in the same sense that I used the word earlier. This leaves the problem, that one cannot expect to identify "hard truths" like in natural sciences, where it is possible to identify natural laws. Studying enterprises gives much room for interpretation. This ambiguousness is also evident in the literature concerning EA, SOA, and BPM. All three concepts lack stringent definitions, and interpretations. Moreover, all three fields are constantly evolving and expanding. A principally important hermeneutic question is to understand, whether there is only a single legitimate interpretation, or if there is in fact multiple legitimate interpretations. [Kvale:208] The obvious answer must be that there are several legitimate interpretations, since all concepts are broad, and are applied in different ways.⁵ Because there are no commonly accepted way of understanding either of these three concepts, my descriptions and definitions of these terms, becomes as much my own interpretation of them.

⁵ Which does not eliminate the possibility that some definitions can be considered wrong or illegitimate

The hermeneutic circle is an approach that can be used to reduce the problem of “one-sided” interpretations, and I will embed the hermeneutic circle in my work. The hermeneutic circle states, that understanding the parts cannot be done without referencing the whole, and understanding the whole cannot be done without referencing the parts. Although sounding like a paradox, the idea is rather simple; there is a continuous shift between looking at the parts, and looking at the whole. I start with a pre-understanding of the subject matter. From there, I will start by understanding the parts by performing literature study, and from this further understand the whole. Once the whole has been grasped, I will return to revise my understanding of the parts. This circle of interpretation does in principle go on continuously, but in reality, the circle will stop, once an acceptable level of understanding has been achieved. [Kvale:57] By adopting the hermeneutic circle, I will progressively arrive at an interpretation more free from bias.

To further strengthen the quest for objectivity, I will perform a range of interviews with people that I consider experts within relevant domains (see appendix C for a list of interviewees). The purpose of these interviews are two-fold; they are meant to help me improve on my body of knowledge, but they are also meant to help interpret the knowledge obtained in the theoretical literature study. Thus, these interviews becomes a part of the hermeneutic circle, by letting them impact my interpretation of the literature.

1.6 Thesis structure

Module 1 - Background and concepts

The first module lays the theoretical foundation for the rest of the thesis. Many of the key terms relevant for this thesis, have different meanings depending on context. It is therefore necessary to describe and define the key terms, that will be used throughout this thesis.

- **Chapter 2:** This chapter concerns enterprise agility. Agility is a phrase gaining much attention, and it is vital to gain an understanding of the meaning of agility, especially seen in relation to business processes and information systems. The key objectives will be to understand the meaning of the term agility, and to understand how agility concept interfaces with other important concepts.
- **Chapter 3:** This chapter serves to establish a conceptual understanding of EA, SOA and BPM as individual parts. Each of these strategic initiatives has the potential to contribute to enterprise agility in certain ways. Understanding the parts, gives us basis for analysing the sum of the parts in the next module of the thesis.

Module 2 - Understanding the sum of the parts

In this module, the sum of the parts will be analysed. A basic premise of this thesis is that EA, SOA and BPM are similar in certain ways and different in certain ways, which allows enterprises to use them in a complementary fashion. Thus, the goal of this module is to investigate how a more integrated approach to EA, SOA and BPM can improve agility.

- **Chapter 4:** The first step of the analysis, is to look at the relationship between BPM and SOA. The vision of BPM and SOA together will be presented. From there, I will analyse the nature of the relationship between BPM and SOA. I will perform this analysis by seeing BPM-SOA from two different perspectives, i.e. in the form of a model-driven perspective and a methodology-driven perspective.

- **Chapter 5:** Having established the relationship between BPM and SOA, it will be argued that EA provides key tools in establishing and maintaining BPM-SOA. The analysis will especially centered around the relationship between alignment and agility. At the end of the chapter, the integrated model will be presented. But I will also identify missing parts to this integrated approach and I will see a shared language to be used across architecture artefacts as a necessity for integrating EA, SOA and BPM for agility. I will set forth the requirements for such a language.

Module 3 – Connecting the Dots

This module is concerned with connecting all the dots of this thesis. The module will consist of two chapters; in the first chapter I will lay the theoretical foundation for a way to implement a shared language for EA, SOA and BPM. The second chapter will consist of a case scenario, including a demonstration of an implementation of a shared language.

- **Chapter 6:** In this chapter, I will lay the theoretical foundation for an implementation of a shared language for EA, SOA and BPM. I will introduce the concept of ontologies, which will be seen as a logical progression from taxonomies. Whereas taxonomies are hierarchical classifications schemes, ontologies are suited for drawing the relationships between thousands of components in an architecture.
- **Chapter 7:** This chapter will codify the important findings of this chapter. The chapter will also include a demonstration of the shared language, based on the requirements set forth in chapter 5.

Module 4 – Conclusion, Perspectives and Criticism

The final section of the thesis will contain the conclusion, perspectives on the thesis subject as well as critical self-reflection on the thesis research.

- **Chapter 8:** Conclusion, perspectives and criticism

Module 1 – Background and Concepts

Chapter 2 – Enterprise Agility

The purpose of this chapter is to set the scene for the rest of this thesis, as it creates an understanding of the term *enterprise agility*. The first step will be to understand the term *enterprise*, and then proceed to a definition of *enterprise agility*. The description of agility, should especially be seen in the light of the thesis topic, focusing on business processes and information systems. Following the definition of agility, I will proceed to discuss how agility interfaces with other important concepts.

2.1 Definition of Enterprise

The word enterprise is used in many different contexts, and usually refers to some form of organisational entity. The term is also sometimes used to denote server-side, or serious industrial software. But typically it is implied, that an enterprise represents some kind of organisational entity, company, or a business. The term *enterprise* have been chosen for this thesis for a particular reason; the use of the word enterprise indicates, that both private and public sector entities are a subject of study. While there are obvious differences between private and public enterprises, there are also a growing number of similarities. Public enterprises are increasingly subject to new demands, such as increased privatisation, benchmarking against private market players, outsourcing or sourcing of activities, as well as reorganisations and structural reforms. In this sense, they have increasingly similar requirements for agility, as their private market counterparts.

A definition of the enterprise term is put forward by Scott Bernard:

"An area of common activity and goals within an organization or between several organizations, where information and other resources are exchanged" [Bernard:31]

From this definition it can be discerned, that an enterprise can be a part of a larger organisation, span the entire organisation, or span across several organisations. Exchange of information and other resources, defines the boundary of the enterprise.

This is contrasted by the definition from TOGAF:

“A good definition of ‘enterprise’ in this context is any collection of organizations that has a common set of goals and/or a single bottom line. In that sense, an enterprise can be a government agency, a whole corporation, a division of a corporation, a single department, or a chain of geographically distant organizations linked together by common ownership.”

[TOGAF:4]

The two definitions have a lot in common. Again, we see that an enterprise can be part of a larger organisation, span the entire organisation or span several organisations. The difference is that Bernard sees exchange of information and other resources as the boundary, whereas TOGAF sees joint ownership as the boundary.

This difference in definition is subtle, but not entirely without importance. The boundaries of the enterprise is getting ever more permeable; enterprises are sometimes considered *extended*, because they include partners, customers and suppliers. [TOGAF:4] Large corporations or governments agencies, may also span several enterprises [TOGAF:4] For all practical purposes, I do however consider this distinction of no particular relevance to this thesis.

2.2 Definition of Enterprise Agility

A simple dictionary definition of agile, would be something like: *“able to move quickly and easily”*. Some of the words we typically associate with being agile are: *quick, fast, nimble, flexible*, and *able to change direction fast*. The different words in use suggests us, that agility is not a one-dimensional construct. The dictionary definition of being able to move quick and easily, would however not constitute an appropriate definition of what it means to be an agile enterprise.

Gartner has developed a useful definition of enterprise agility: *"the ability of an organization to sense environmental change and respond efficiently and effectively to that change"* [Gartner, 2006:2] Another view is presented by Vince Kellen, who defines strategic agility as: *"a firm's ability to successfully handle significant, fast-moving, and frequent unknown events within its competitive space"* [Kellen:2] It is obvious that the Gartner definition and the definition of strategic agility provided by Vince Kellen resembles each other. Both definitions implies, that agility is not only about speed or flexibility. Gartner talks about responding *efficiently and effectively* to change, whereas Kellen uses the word *successfully*. There are two important observations. Firstly, we see that they are both talking about responding to change in the external domain, either in the form of environmental change or changes in the competitive space. Secondly, both of them are reactive, rather than proactive; agility is thus about responding to changes. In reality, we can consider the two definitions more or less equivalent except for one thing; Gartner also emphasises the ability to sense environmental change. This is similar to the notion of *early warning agility* mentioned in Rambølls *"IT i Praksis"* report.⁶ [Rambøll:6]

IT plays a special role in regards to agility, and Kellen adds another layer of agility to the discussion. He defines IT strategic agility as: *"a subset of strategic agility and refers to specific IT resources and capabilities that need to contribute to the firm's overall agility"*. [Kellen:2] The notion of IT strategic agility as a subset of strategic agility is an important one; an enterprise does not become agile, just because it deploys a flexible IT-architecture.⁷ Enterprises becomes agile when they are able to adapt to changes in the environment, in similar ways to biologic organisms, who would need to adapt to a changing environment.

6 Using the term agility isolated about the capability to detect changes in the environment, does seem like an overuse of the word agility. Being able to detect changes in the environment is an important part of being an agile enterprise, but this capability does not in itself produce a response to the changes. *Early warning capability* would in my mind have been a better term.

7 This will be an important discussion later on. But let me already now indicate, that I think one of the fallacies of SOA is to believe, that SOA in itself makes an enterprise more agile.

To further clarify the way I interpret agility in regards to this thesis, I have created a conceptual life-cycle model for enterprise changes. The life-cycle assumes that an enterprise detect a change in the environment, and then decides to react to this changes, by creating/modifying business processes or IT-capabilities. The model is purely illustrative, as I am not suggesting that the process of change necessarily goes through a series of discrete steps (nor that it should).

Phase	Description
Detect	The enterprise detects changes in the environment and makes an estimate on the potential impact of this change. If the enterprise perceives the change in the environment to be an opportunity or a threat, it decides to goes forward to the analysis.
Analysis	During the analysis the enterprise seeks to understand the problem, and to develop the business requirements for a solution.
Design	Based on the requirements, the enterprise develops conceptual solutions to the problems. The solution might involve changed/new business processes and/or changed/new IT-capabilities
Development	Business processes and IT-capabilities are developed, using the development paradigm of choice in the enterprise.
Deploy	New or changed business processes are deployed in unison ⁸ with new or changed IT-capabilities.

Table 1: Enterprise Change Process [Own production]

In the context of this thesis, I am going to view agility in a particular way; agility is defined as the capability to effectively and efficiently manage business changes, manifesting itself in changes to business processes and their underlying information system. This means that changes should be managed across all these life-cycle phases.

⁸ I am not suggesting, that just because business and IT act in unison, they are also agile. See the next section on Agility versus Alignment for more about this subject

Among drivers for business changes would be:⁹ [TOGAF:112-113]

- Business-as-usual developments
- Business exceptions
- Business innovations
- Business technology innovations
- Strategic change

As can be seen, this definition does not seek to define, whether these changes are driven by strategic or tactical concerns. Rather, this definition stems from the observation, that business processes are important assets, both on the strategic and tactical level.

2.3 Enterprise Agility and Information Technology

With changing business processes, and their related IT systems as the focal point of this thesis, it is important to consider the relationship between IT and agility. The focus on agility does not only depart from the observation, that the environment in which enterprises are operating, has become more complex, more competitive, and more fast paced. Rather, agility is becoming an important subject in part because enterprise information systems have shown to be remarkably rigid to change. This dichotomy between environmental factors shaping the need for agility, compared to the relative rigidity of information systems is highlighted by the following quote:

“A recent survey of Fortune 500 companies indicated that over 80% had altered their business model in a given two-year period. Two thirds of these – roughly half of the total respondents – claimed that this business change had been constrained by inflexible IT. In a survey by IBM Business Consulting Services, 90% of CEOs expect to transform their enterprise to become more responsive, particularly to customer demand, within the next five years” [Sprott:3]

⁹ The drivers for changes mentioned here, has been taken from the TOGAF “Architecture Change Management”. There are of course also drivers for technical changes, such as new technology developments. These are not deemed relevant to this thesis.

So the relationship between agility and IT can be seen from two opposite directions; either as enablers of agility, or as impediments to agility. As enablers of agility, modern information systems affects agility positively in three ways: [Huang & Nof:51]

- **Speed:** By speeding up activities in the enterprise, f. ex. by the use of automation
- **Decision support:** By providing intelligent and autonomous decision making processes
- **Distribution:** By enabling distributed operations with collaboration

[Bloomberg & Schmelzer:13-14] however points out, three ways in which IT becomes impediments to agility.

- **Complexity:** Modern enterprises are very complex entities, shaped through organic growth, mergers, and acquisitions. This complexity is also found on the technology side, where enterprises must cope with infrastructure layers piled upon layers, often without any overarching architectural considerations. Complexity is a major impediment to agility, as it forces enterprises to spend inordinate amount of resources in order to figure out the existing infrastructure, and to devise new solutions.
- **Inflexibility:** Inflexibility is the result of complexity. If processes or systems are challenging to change, enterprises are likely to be very cautious about making changes. Inflexibility is clearly an impediment to agility here; if you cannot change your IT-systems, then it is very hard to adapt to the changed environment. Inflexibility often leads to a "*if it ain't broken, don't fix it*" mentality.
- **Brittleness:** The other side of inflexibility is brittleness, which refers to how fault tolerant something is. Brittle architectures are characterised by being prone to breakdowns, when small changes are introduced to the architecture. So if tiny changes can produce huge negative side-effects, then the architecture would be considered brittle.

2.4 Agility versus Alignment

The relationship between agility and alignment is however not always clear. As [Bloomberg & Schmelzer:13] points out, alignment is sometimes being mistaken for agility. The central tenet of alignment is that effectiveness is not obtained by focusing on individual components. Such an approach would lead to sub-optimisation, rather than the creation of an overall effective enterprise. It can be said, that *“effectiveness is driven by the relationship between components, rather than by detailed specification of each component”*. [Lankhort:6]

The perhaps most famous model for alignment is the *Strategic Alignment Model* by Henderson and Venkatraman. An adapted version of the model is depicted in figure 1.

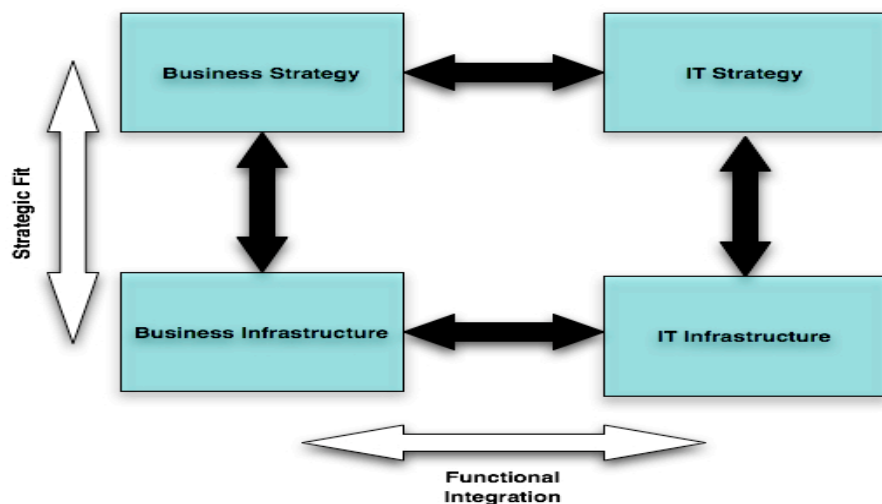


Figure 1: Strategic Alignment Model [Adapted from Venkatraman & Henderson:479]

The models depicts two different dimensions of alignment, where the Y-axis concerns the *strategic fit*, i.e. the fit between the external external and internal domains of the firm. [Henderson & Venkatraman:472-473] The external domain concerns the positioning and differentiation of the firm within the marketplace, whereas the internal domain in contrast, concerns the interior of the firm, including but not limited to the design and redesign of critical business processes. The authors not only argues, that the strategic fit between the external domain and the internal arrangements of the firm is key to maximising economic performance, but also that the strategic fit should be on done both the business and IT domains.

The second dimension concerns functional integration, otherwise known as business-IT. Business-IT alignment. Alignment between business and IT is perhaps the holy grail of IT management, and much research has been done in this area.¹⁰ Although not always consistently, business-IT alignment is usually linked to improved performance, and improved leverage of IT investments. It is sometimes argued, that the inability to realise value from IT-investments is at least in part due to lack of alignment between business and IT. [Henderson & Venkatraman:472] Empirical studies seems to support this notion. F. ex. in a study among 84 pairs of plant managers, and IT managers it was found, that there was a synergistic relationship between strategic alignment and IT investments; firms with poor alignment can improve profits and revenues without investing more in IT, just by improving business-to-IT alignment. Well-aligned firms can increase profits and revenues, by increasing IT-investments, knowing that the increased investments are likely to pay off. [Byrd, Lewis & Bryan] Strategic alignment may however not provide the above cited benefits. According to [Chan et al.] creating the IT strategy is only a first step, as vital as it may be. Rather, the realised IT¹¹ strategy is important, suggesting that making joint plans between business and IT is not the same as actually fulfilling these plans.¹² [Henderson & Venkatraman:472] holds a similar view, and sees strategy as as not only formulation, but also implementation of the strategy.

10 [Chan & Reich] f. ex. provides an annotated biography of more than 150 IT-alignment related articles

11 The authors of the article uses the word IS instead of IT. To maintain consistency in this thesis, I have substituted the authors use of IS with IT, knowing that some degree of approximation can be made between the two terms.

12 With the observation that intended use of technology often differs from actual use, [Chan, Huff & Copeland] further argues, that having a documented business or IT strategy, is not the same as having the capabilities to implement said strategies.

Alignment and agility are clearly not the same concepts, but what is the relationship then between them? It is obvious, that we can have agility without alignment, as well as alignment without agility. Agility without alignment, would at least in the short run lead to an enterprise responsive to changes, as it would be possible to make ad hoc changes rapidly. It would however also be rather inefficient in the long run, and would lead us back the road where we came from; towards a disorganised architecture that becomes complex, inflexible, and brittle. Eventually, this would become an impediment to agility in itself. On the other hand, we can clearly also have a perfectly aligned enterprise, which is very unresponsive to change. Such an enterprise would be efficient, but would not be sustainable in a volatile environment. So alignment and agility can certainly be opposed to each other.

But the opposite can also be true; alignment and agility can in some way be supportive of each other. A more flexible IT-infrastructure would make it easier to perform the necessary steps in order to achieve alignment. Alignment on the other hand, is important for creating an architecture that is flexible and maintainable in the long run. This should be compared to that of ending up with overly complex architectures, emerged from continuous short-term decision-making about the architecture. These are important points to keep in mind in regards to this thesis; the concept of agility cannot stand alone. In the long run, there is likely a relationship between alignment and agility. This means, that even if agility is the primary goal in this thesis, alignment will have to play a secondary role. I will touch further upon the topic of alignment in later chapters.

2.5 Enterprise Agility and Agile Software Development

Some practitioners of IT would likely ask about the relation between enterprise agility, and the agile software development methods. The agile software development movement covers a range of various flavours of lightweight, and iterative development methods, including but not limited to XP, Scrum, and RUP. The movement is epitomised by the *Manifesto for Agile Software Development*, which is centred around a collection of software development principles. [Manifesto]

There are some reasons why agile software development does not play a role in this thesis. The agile movement is primarily seeking to address the problem of inherent *requirements unpredictability*, i.e. that requirements are changing late into the project. By introducing the notion of iterations, agile methods are more adaptable to requirement changes during the development process [Fowler, 2005] This is in itself a laudable goal, and an agile approach to software development might contribute to improve IT strategic agility. But it does not address the problem of how to design the business, and how to develop the right business requirements in the first place. The business modelling¹³ taking place in agile development methods are used for understanding the problem domain; not for developing the right business requirements. In RUP f. ex. the purpose of business modelling is to “*understand and communicate the structure and the dynamics of the organisations in which the system is to be deployed*”. [Larman:483-484]

Because of this, the agile development paradigm can be considered primarily an operational initiative, focusing on improving the efficiency of the software development process. This may impact IT strategic agility, but the effect on strategic agility is more indirect. For the remainder of this thesis, the words *agile* and *agility* will not refer to the agile software movement, unless stated otherwise.

13 Or domain modelling

2.6 Chapter Summary

The purpose of this chapter was to lay the foundation for the rest of the thesis. The chapter begun with a definition of the term enterprise agility. By comparing different definitions, it was found that enterprise agility is generally an ability to react efficiently and effectively to changes in the environment. This makes agility a reactive concept. The notion of early warning capability was also stressed.

In line with the scope for this thesis, agility was defined more precisely as the capability to effectively and efficiently manage changes in business processes and their underlying information system. Such business changes can be driven by either strategic or tactical concerns. The problem of complex, inflexible, and brittle IT-architectures, as an impediment to agility was briefly discussed.

Some energy was spent on discussing the relationship between agility and alignment. Agility and alignment are definitely two different concepts, but their relationship is not perfectly clear. On one hand, agility and alignment can be polar opposites. On the other hand, the two concepts can go together, and be supportive of each other. Alignment will be an important concept in later chapters.

Finally, the agile software movement was discussed, and it was found, that it was primarily an operational initiative, focusing on improving software development efficiency. As such, it had no direct impact on agility, and is thus not a further topic for study in this thesis.

Chapter 3 – Understanding the parts

The purpose of this chapter is to establish a common understanding of each of the three main strategic initiatives. Understanding each of these, as individual parts, will provide the basis for analysing the relationship between them in later chapters.

3.1 Enterprise Architecture

People are familiar with using the word *architecture* when thinking about buildings and other large man-made physical structures. Encyclopædia Britannica says about architecture: *“the art and technique of designing and building, as distinguished from the skills associated with construction”*. [Britannica] But architecture does not just concern large man-made structures. Software architecture is a commonly known example of architecture applied to information technology. IEEE 1471:2000 is a recommendation for describing the architecture of software-intensive systems. Such a system is:

“any system where software contributes essential influences to the design, construction, deployment, and evolution of the system as a whole.” [IEEE1471:1]

The term *architecture* is increasingly applied in broader contexts, such as on enterprise level. It is worth noting, that the IEEE 1471:2000 has been adopted as ISO standard ISO/IEC 42010:2007. A joint revision between IEEE and ISO is underway in order to widen the scope to general systems architecture, which will include enterprise architecture. [ISO42010] Architecture itself can be defined as:

“The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.” [IEEE1471:3]

This definition of architecture will be of great importance in later chapters.

3.1.1 What is Enterprise Architecture?

The term *Enterprise Architecture* (EA) was probably used the first time by Steven Spewak in his book “*Enterprise Architecture Planning*”. [Bernard:32] Following the landmark article “*A Framework for information systems architecture*” by John Zachman in 1987, EA has widely become associated with him. His framework was extended in 1992 in collaboration with J. F. Sowa, [Zachman & Sowa] and has been revised several times since then. In his original article Zachman observed that:

“With increasing size and complexity of the implementations of information systems, it is necessary to use some logical construct (or architecture) for defining and controlling the interfaces and the integration of all of the components of the system”.

[Zachman, 1987:276]

In this sense, enterprise architecture emerged as an enterprise-wide approach to manage the complexity of information systems architecture. Zachman did deliberately abstain from pursuing the relationship between building the business strategy, and the linkage to information system strategy. Zachman wrote:

“The development of a business strategy and its linkage to information systems strategies, which ultimately manifest themselves in architectural expression, is an important subject to pursue; but it is quite independent of the subject of this work, which is defining a framework or information systems architecture” [Zachman,1987:277]

The enterprise-wide view is perhaps the most salient feature of EA. But the general notion of EA has however changed since Zachman published his article in 1987. EA is naturally still positioned as enterprise-wide, but has become more holistic. Contemporary definitions of EA now also includes elements such as strategy and business architecture.

Some common definitions of EA are:

- **The Open Group:** *"Enterprise architecture is about understanding all of the different elements that go to make up the enterprise and how those elements inter-relate."* [Schekkerman:21]
- **Mark Lankhorst:** *"Enterprise architecture: a coherent whole of principles, methods, and models that are used in the design and realisation of an enterprise's organisational structure, business processes, information systems, and infrastructure"* [Lankhorst et al.:3]
- **Scott Bernard:** *"The analysis and documentation of an enterprise in its current and future states from an integrated strategy, business and technology perspective"* [Bernard:31]

The definitions are all different, but there are some commonalities. Based on the different EA definitions, some common traits of modern EA can be identified:

- **Enterprise-wide:** EA programs spans the entire enterprise and provides a way of thinking about resource utilisation that spans across the entire enterprise. ¹⁴
- **Holistic:** EA is seen as something that help understands the individual elements making up an enterprise, and how they relate to each other.¹⁵ The elements that makes up the enterprise, often includes both strategy, business and technology.
- **Documentation-centric:** All flavours of EA relies on architecture descriptions for documenting the enterprise. ¹⁶
- **Transformation-oriented:** The purpose of EA is to somehow change transform the enterprise. By documenting the enterprise in the current *as-is* state and a desired *to-be* state, it is possible to devise a transformation plan

¹⁴ Many frameworks do however support the notion of segmentation, so that individual parts of the enterprise can be treated as slices of the entire enterprise.

¹⁵ Which is similar to the definition of architecture given in IEEE:1471:2000

¹⁶ The documentation can either be in the form of model artefacts or non-model artefacts. Models will become an extremely important topic later in this thesis.

By taking an enterprise-wide and holistic perspective on the enterprise, it should be possible to achieve functional integration between business and IT. And often Business-IT alignment is hailed as one of the primary internal drivers behind the adoption of EA.¹⁷ [Lankhorst et al.:6] EA is a tool that can be used to escape the bottom-up systems thinking that has dominated IT-development the last couple of decades. As Scott Bernard points out, the bottom-up thinking has led to stove-pipe applications and duplicative efforts. [Bernard:63] Other often cited benefits of doing EA are increased agility, [Bernard:63] reduced exposure to regulatory and compliance risks, [Lankhorst et al.:8-10] and reductions in cost and complexity. [Schekkerman:15]

3.1.2 Elements of Enterprise Architecture

In the following I will give an overview of the core terminology of EA. I will rely on the definitions, as given by Scott Bernard in this “*An Introduction to Enterprise Architecture*”, because his terminology is fairly consistent and comprehensive. The main concepts are:

- **EA framework:** The first element is the EA framework, which serves as a structure for organising EA artefacts. As such, it both defines the scope of the documentation effort, as well as describing how the different areas of the architecture relates to each other. [Bernard:81] An EA framework can be considered a taxonomy, as taxonomies are used to classify entities within a certain domain. [Gasevic, Djuric & Devedzic:52]
- **EA methodology:** A step by step description of how EA will be implemented and how the documentation will be developed, archived and used. The methodology includes selection of framework, modelling tools, and on-line repository. [Bernard:81]
- **EA artefacts:** These are not in themselves part of EA, but are documentation products from the EA program. [Bernard:87] EA artefacts f. ex. concerns the description of the enterprise in as-is and to-be states. Artefacts also includes principles, standards, reference-models etc.
- **EA tools (including repository):** Used to persist and manage artefacts produced during the EA program.

¹⁷ In this case, as in most others, alignment primarily means functional integration, and not the creation of strategic fit

3.1.3 Enterprise Architecture Frameworks

There are many different ways to approach EA. In the book “*How to survive in the jungle of Enterprise Architecture Frameworks*”, the author describes more than 10 different EA frameworks. [Schekkerman] This is even without counting the many frameworks created by EA tool vendors, consultancies, or by enterprises themselves, including but not limited to those related to governmental EA programs. The huge number of approaches to EA is perhaps not surprising, given how complex and different enterprises can be. But the many different takes on EA does however give us some problems navigating the terminology. The probably most confusing concept in EA is the use of the word *framework*. In general, different flavors of EA are referred to as EA frameworks. But as it was seen from Scott Bernard's terminology given earlier, EA does in itself contain a framework. For the remainder of this thesis, I will use the term EA Framework to indicate a certain way to do EA. Whenever I refer to a framework in the meaning of a taxonomy, I will use the term *taxonomic framework*.¹⁸

In the following, the Zachman Framework and The Open Group Architecture Framework (TOGAF) will be described.¹⁹ There are a couple of reasons for this. Firstly, both frameworks are very popular takes on EA. Secondly, each of them represents complete different angles on EA. By mainly serving as a tool for the classification of architecture descriptions, the Zachman Framework is a taxonomic framework in its purest sense. Moreover, the Zachman Framework does not have any methodology associated. The TOGAF in contrast, clearly has the methodology as focal point. Moreover, TOGAF does not in itself contain a taxonomic framework.²⁰ The two frameworks can in this sense be considered somewhat complementary.

¹⁸ Unless I directly refer to the Zachman Framework

¹⁹ Version 8.1.1, Enterprise Edition

²⁰ This statement will be modified a bit later, as the Enterprise Continuum of TOGAF is discussed.

3.1.3.1 The Zachman Framework

The Zachman framework is a classic taxonomic framework, serving as a structure for defining the scope of the enterprise architecture, as well as a structure for classifying EA artefacts. The framework can be seen depicted in figure 2 below. The framework can be found in full size in appendix D.

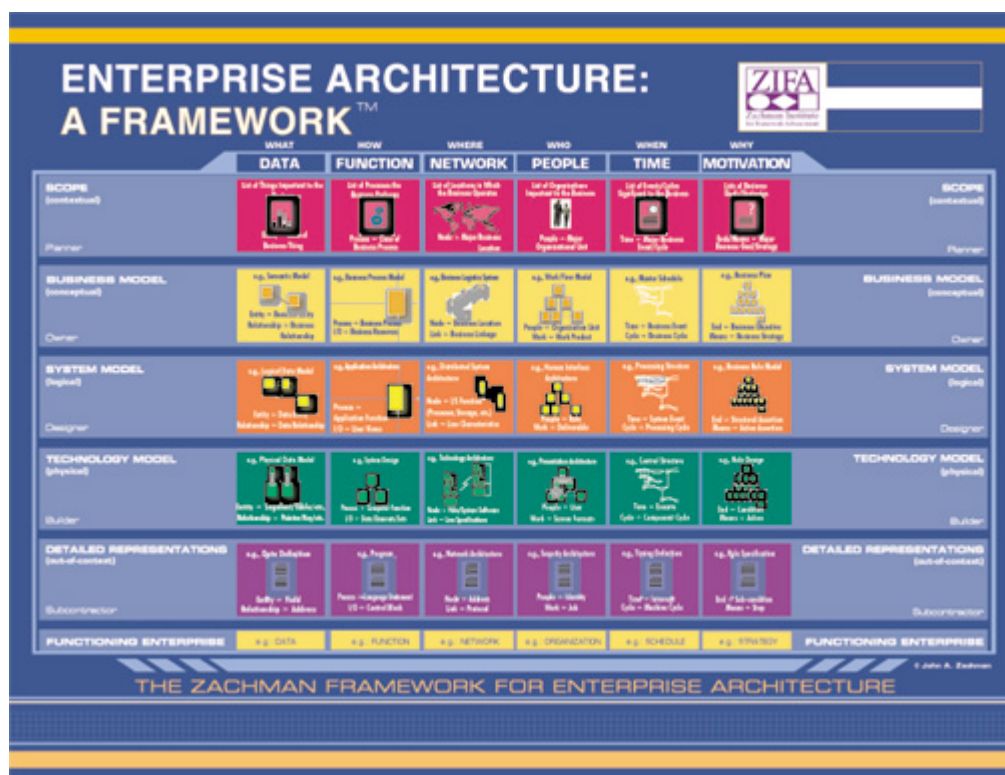


Figure 2: The Zachman Framework [ZIFA]

The framework itself, is a matrix structure with two dimensions. Each cell in the matrix is a normalised aspect of the enterprise. On the vertical axis, the framework is partitioned into six different layers, showing the enterprise at various levels of abstraction. These layers, often called perspectives, consists of different types of models, such as business models, logical models, and physical models. The idea behind the use of different perspectives is that different types of stake-holders sees the enterprise in different ways. A holistic view of the enterprise can however only be achieved through looking at all of the perspectives.

The perspectives are:

- **The Scope (Contextual):** Aimed at the planner
- **The Business Model (Conceptual):** Aimed at the owner
- **The System (Logical):** Aimed at the designer
- **The Technology (Physical):** Aimed at the builder
- **The Detailed Representations (Out-of-Context):** Aimed at the subcontractor
- **The Functioning Enterprise viewpoint:** The actual running systems or functions

On the horizontal axis, the framework contains a classification of artefacts, i.e. it serves as a taxonomy for classifying concerns. [TOGAF:465] The horizontal axis contains a range of aspects, expressed in the form of *wh*-questions, following the classical "who-what-where-when-why-how" pattern.²¹

- **The Data aspect:** What?
- **The Function aspect:** How?
- **The Network aspect:** Where?
- **The People aspect:** Who?
- **The Time aspect:** When?
- **The Motivation aspect:** Why?

The main strength of the Zachman Framework, stems from it being a taxonomy for enterprise architecture descriptions. As such, the framework is indeed a very comprehensive taxonomy for models and other artefacts [TOGAF:465] Since a good taxonomy is characterised by being able to "*separate its corresponding entities into mutually exclusive, unambiguous groups and subgroups that, taken together, include all possibilities*" the framework can be considered a good taxonomy.²² [Gasevic, Djuric & Devedzic:52]

²¹ With the notable exception of "How"

²² Perhaps with the exception, that the Zachman Framework does not attempt to specify any cells in relation to security, nor manageability.

It should however be noted, that as influential the framework has been, it has not been free of criticism. The main critical point is that the framework has no methodology attached, and provides little guidance on how to fill out the cells. Furthermore, there are concerns that the amount of documentation required to populate all cells is too high. It could be argued that few (if any) enterprises, are capable of mastering the contents of all 30 cells. The framework should instead be seen as a best-case, and populating the framework should be guided by the needs of the enterprise. [Bloomberg & Schmelzer:123] Zachman himself, however stress the importance of having all cells made explicit, at a high level of detail: [Zachman, 2000:1]

"I am confident that at some point in time, the Enterprise is going to wish it had all of those design artifacts (models, cells of the "Zachman Framework," the Framework for Enterprise Architecture) made explicit, Enterprise-wide, horizontally and vertically integrated at excruciating levels of detail...."

Lastly, a basic problem with the Zachman framework is that it does not describe the enterprise as an integrated whole, because it does not include interaction bonds between the artefacts in different cells. Thus, the enterprise is regarded as an aggregate of artefacts. The Zachman framework originally did not provide much information, as to the relations between cells. Later, Zachman has expanded on his original thinking, and recognises that *"each cell has relationships to other cells in the same row"* [EABOK:30]. Some holds that the Zachman Framework is today regarded more of a thinking tool, rather than being a practical framework for EA. [EABOK:30] In any case, the influence of the Zachman Framework on modern EA should not be dismissed.

3.1.3.2 The Open Group Architecture Framework

TOGAF is emerging as a popular EA framework, and can be considered a very comprehensive methodology for creating architectures. The focal point of TOGAF is to provide a set of “*methods and tools for developing a broad range of IT architectures*” [TOGAF:11]. As the quote indicates, TOGAF is rather technology-centric, and business architecture was not added until version 8 of the Enterprise Edition. [Temnenco] Currently, TOGAF includes four “kinds” of architecture:

- Business Architectures
- Data Architectures
- Application Architectures
- Technical Architectures

The combination of Applications Architecture and Information Architecture is collectively referred to as Information Systems Architecture. [TOGAF:11] TOGAF is intended to be a generic framework, which can be used in a wide variety of environments. It does not prescribe a specific set of deliverables (EA artefacts). Rather, it describes a set of generic deliverables by example. [TOGAF:4] [TOGAF:12] The focus is on the types of deliverables to be produced, and on how to produce them. Enterprises may therefore decide to use the generic TOGAF deliverables, or decide to include a taxonomic framework, containing more precise specifications of the deliverables to produce.

TOGAF consists of three main parts, of which the Architecture Development Method (ADM) is the central one. The ADM is a method, which is hailed as “a *reliable, proven method for developing an enterprise IT architecture that meets the needs of your business*”.²³ [TOGAF:7] The ADM is illustrated in figure 3 (full size image can be seen in appendix E). Other parts of TOGAF are the Enterprise Continuum and the Resource Base. The Enterprise Continuum can be regarded as a virtual repository of all architecture assets [TOGAF:131] As the enterprise works through the ADM process, it will populate its own Enterprise Continuum. [TOGAF:18] TOGAF makes no recommendations as to the selection of a taxonomic framework nor make any recommendations on how to persist, and manage EA artefacts physically. The final part of TOGAF is the Resource Base, which contains a range of resources, including guidelines, templates, checklists, and other detailed materials to be used in conjunction with the ADM. [TOGAF:18].

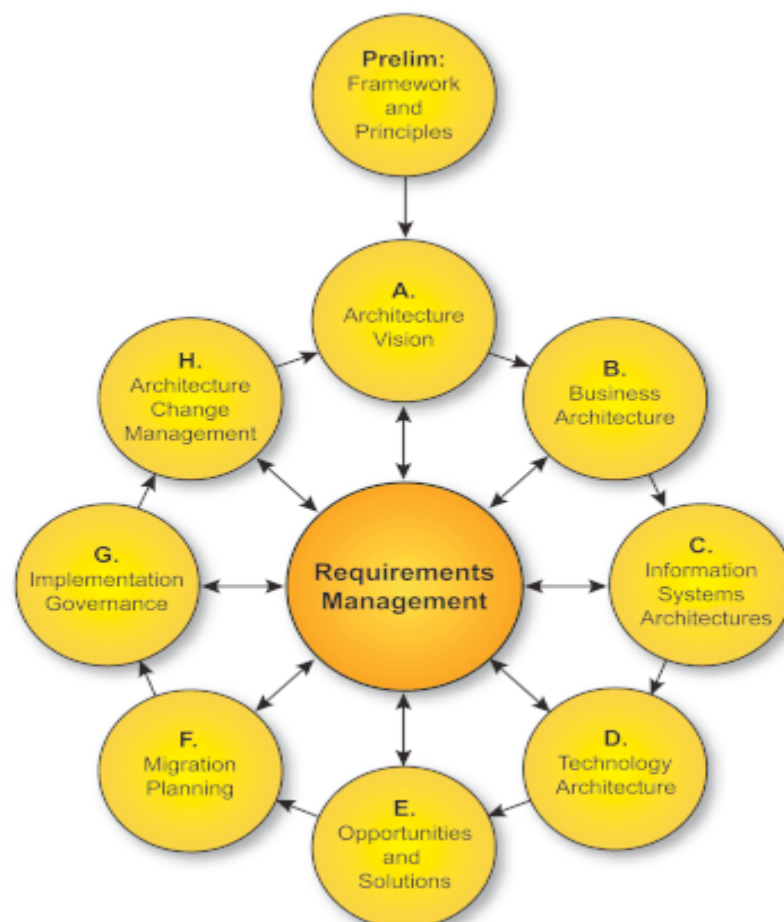


Figure 3: Architecture Development Method
[TOGAF:20]

²³ Again, we see the technology focus in TOGAF; the main point is to develop IT architectures that are aligned with business needs.

The ADM is an iterative process for developing architectures, including nine different phases (see figure 3). The Requirements Management process is at the center of the ADM, and is for identifying, storing, and feeding requirements in and out of relevant ADM phases [TOGAF:18]. Each phase contains descriptions of its objectives, inputs to the phase, outputs of the phase, as well as the steps that needs to be performed during the phase to produce the output.

The overall impression from this description of TOGAF is that TOGAF can be regarded as a very practical approach to architecture development.

3.2 Business Process Management

Being able to manage business processes for efficiency, effectiveness, and agility is a vital competency in ensuring continued survival and thrift of the enterprise. This challenge has not been ignored by CIO's. In fact, the 2007 Gartner EXP survey revealed, that improving business processes was the number one business priority among the 1400 CIO's participating. [Gartner, 2007]

3.2.1 What is Business Process Management?

The first step in understanding BPM will be to understand the role of business processes and why managing them is a challenge. A business process can be defined as:

“A set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships” [Khoshafian, 2007:224]

This definition highlights a central issue; while business processes are certainly important assets to the enterprise, enterprises are not driven by business processes. A process should ideally, always satisfy a goal or a policy for the enterprise. Moreover, business processes are sequences of activities, which essentially are collections of tasks that contributes or adds value to process goals. [Jeston & Nelis, 2008, 1:183]

Another important distinction is to be made; for the remainder of this thesis, I will consider business processes the organising logic for the activities taking place. The organising logic is to be distinguished from the actual underlying work being done (in the activities). The importance of this distinction will become apparent later, when the attention is turned towards the relationship between business processes and services in SOA.

Often, business processes are not the result of design, but tends to emerge over time. [Juric & Pant:11] Business processes are difficult to manage for several reasons. Firstly, managing business processes is complex because business processes are themselves complex; they often span people, departments, organisational layers and information systems. Furthermore, business processes may include complex business logic. The complexity of processes makes them inherently difficult to understand and to manage. BPM can be regarded as a structured approach to managing business processes and usually enterprises are concerned with realising process improvements by optimising the sequence of activities in the process or by automating entire processes, sub-processes or activities. One definition of BPM sounds:

“The achievement of an organization's objectives through the improvement, management and control of essential business processes.” [Jeston & Nellis:11]

The history of process management paradigms is long, and as there have been successes, there have been spectacular failures. Process management can be divided into three distinct waves. [Smith & Finger:18-19] The first wave was the scientific management school, as put forward by Frederick Taylor. This wave took off in the 1920's. The scientific management paradigm was more focused on organising work activities than focused on automation. The second wave, which flourished from the 1980's and onwards, focused on manually reengineering processes, f. ex. in the form of Business Process Reengineering (BPR) or implementations of Enterprise Resource Planning systems (ERP). Both paradigms were attempts at improving processes revolutionary, rather than enabling incremental improvements. Despite the hype and popularity of BPR in its heyday, the reengineering efforts often failed. Estimation of failure rates vary, but are usually around 50-70%. [Fitzgerald & Murphy] The impact of ERP systems as process management initiatives are much harder to gauge; ERP systems do define *“best-practice”* business processes. But after all, ERP systems are as much about managing technology as managing business processes. Empirical studies, such as [Hunton et al.] does however suggest, that adopters of ERP systems performs better than non-adopters. Incidentally, adopters are not performing better than before adoption; rather, non-adopters are performing worse.²⁴[Hunton et al.:181]

²⁴ It is not clear if the improvements has much (if anything) to do with the business processes themselves.

In the last wave, which should be emerging, processes becomes first class citizens. Change is the primary design goal: *“The ability to change is far more prized than the ability to create in the first process”*. [Smith & Finger:18] Based on the lessons learned from previous process management paradigms, modern approaches to process management are less radical and more evolutionary. They place more emphasis on creating the ability to constantly improve processes. The approach to BPM that I will be using in this thesis, will mostly resemble the 3rd wave of business process management. The focal point will be to enable continuous process improvements, rather than seeking radical re-design of processes.

3.2.2 The Business Process Life-Cycle

One central tenet of BPM is that business processes are having a life-cycle, and that management of processes must occur across all these phases. Figure 5 depicts a conceptual life-cycle model for business processes. As the figure indicates, managing business processes is an ongoing and iterative effort.



Figure 4: Process Life-Cycle [SAP]

The life-cycle model contains four stages:

- **Analysis:** The “business analyst”²⁵ performs analysis of the business process. The analysis can be initiated f. ex. on basis of a process improvement project or because process failures have been detected. In this phase, the business processes are modelled *as-is*.
- **Design:** Through the use of user-friendly modelling tools, the business analyst can design new business processes or make adaptations to existing processes. In this phase, the business process is modelled in a desired *to-be* state.
- **Deployment:** The designed or changed business process is deployed. The deployment may include changes to information systems.
- **Monitoring/Run:** Once the process has been deployed and active, the process is executed. monitored in order to investigate performance, and detect process failures.

Business Activity Monitoring (BAM) solutions can be used to provide an overview of processes within the enterprise [Juric & Pant:44] A BAM solution would f. ex. help gather information about the processes, such as the time to complete different activities, the number of process instances running at any given time or how much time it takes to complete a process. [Juric & Pant:94] Although BAM solutions can indeed provide very valuable information in relation to monitoring and optimisation of business processes, dealing with BAM is out of the scope for this thesis.

²⁵ The term 'business analyst' is frequently used to denote the person that is responsible for business analysis and design. In reality, the 'business analyst' function is likely to be spread across several business stake-holders. For the remainder of the thesis I will use the term 'business analyst' as a 'persona' describing a functional role rather than a person.

3.2.3 Digitisation of Business Processes Management

BPM can and will be done without the assistance of technology. But very often technology will be involved in order to digitise processes. Digitisation of processes can denote two things:

- **Automation of Business Processes:** Digitising business processes involves automating processes, sub-processes or activities
- **Business Process Management Digitisation:** BPM is in itself a process that can be digitised. This involves using digitised *as-is* and *to-be* models to help closing the gap between modelling and execution.

The importance of automating manual work is often overemphasised in relation to BPM. While automation may be valuable because it improves efficiency, automating a fundamentally flawed process provides little value. Effectiveness is not realised unless the process is streamlined to support the goals or policies of the process. The role of digitising the business process management process itself is to facilitate better management of the rest of the business processes.

A Business Process Management System²⁶ (BPMS) is often a vital component in digitising processes. A BPMS is both a software platform and a product category. [Khoshafian, 2007:229] The purpose of BPM Suites are to facilitate the management of the processes digitally. Ideally processes could be managed across all phases of the life-cycle, i.e. it is not only possible to model and design and monitor business processes, but also to deploy (and execute) the processes on the underlying execution platform. [Khoshafian, 2007:231] The ability to manage digitised business processes across the entire life-cycle, will be an important discussion point in the next chapter.

²⁶ Sometimes also called a Business Process Management Suite

It should be noted that the term BPMS is sometimes being equated with BPM itself. In [Khoshafian, 2005] the author f. ex. explicitly equates BPM with BPMS, claiming that it is similar to Database Management and Database Management Systems, where “*in most contexts they are synonyms*”. [Khoshafian, 2005:106] I do however think this is missing the point; BPM is goal-oriented behaviour, i.e. seeking to manage business processes in order to achieve organisational goals. A BPMS is just a tool for realising this. For the remainder of this thesis, the definitions of BPM and BPMS will be kept separate; BPM is methodology, and a BPMS is a tool.

3.2.4 Elements of Business Process Management

Just like their EA counterparts, BPM efforts needs structured methodologies to be used as guidance. BPM has however yet to mature into a structured and acknowledged discipline. [Jeston & Nelis, 2008, 2:1] This means, that BPM often gets used as a catch-all phrase for anything somewhat related to business processes. Moreover, the lack of maturity often means that approaches to BPM are developed from scratch. [Jeston & Nelis, 2008, 2:1] This makes it difficult to deal with BPM at an abstract level. But to be able to understand how BPM relates to SOA and EA later in this thesis, an understanding of the basic elements that makes up BPM is needed. A brief discussion of the 7FE BPM framework will therefore be given below. Figure 5 shows the basic structure of the 7FE framework. The description will be given to illustrate certain characteristics that can be considered generic across most BPM frameworks. Moreover, since the BPM framework is very comprehensive, the description will be kept to the absolute minimum of details that are considered relevant to the rest of this thesis.

The basic elements of the 7FE approach will be described below the figure.

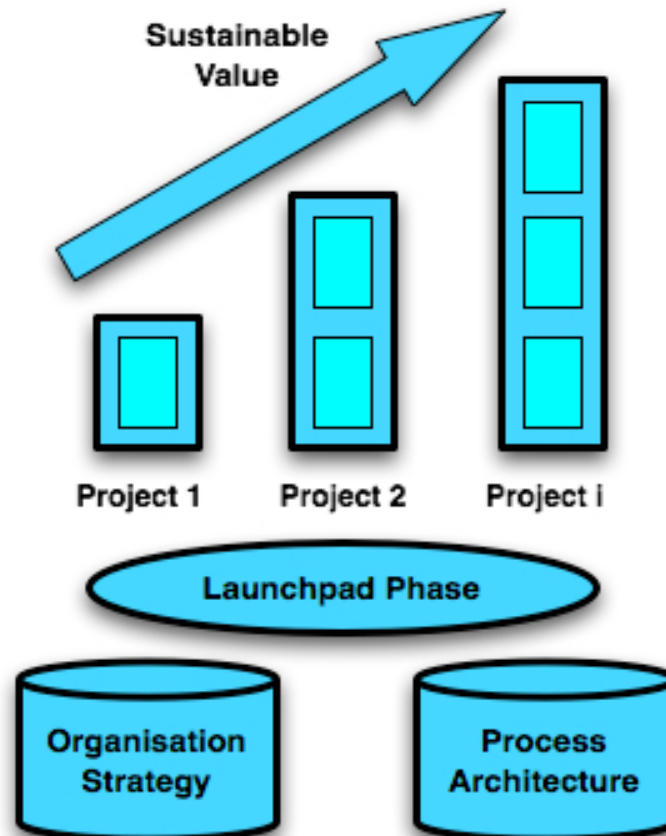


Figure 5: 7FE BPM Framework [Adapted from Jeston & Nellis, 2008, 1:63]

Organisation strategy: As emphasised earlier, processes are not ends in themselves, but rather means through which goals and policies are satisfied. Based on strategy, the management determines goals and policies. Failure to align processes with goals and policies leads to sub-optimal processes.

Process Architecture: Provides the link between organisational strategy and the launchpad phase and forms the foundation for later phases. In particular the process architecture includes:

- **Process guidelines:** General principles to the process domain. Includes standards, methods, guidelines, policies tool selections.
- **Process Models:** A high-level overview of the processes in the organisation. Includes visual representations of high-level processes, links between the processes and a list of end-to-end processes.

The Launch-pad Phase It is rather unlikely, that an enterprise will attempt to manage all of its business processes at the same time. Some processes are more valuable than others, and some processes are more ripe for management than others. In a structured approach to BPM, an enterprise will adopt a formal decision making process to be used as guidance for the identification, prioritisation and selection of processes to manage.

Program/Project phase: BPM is realised through one or more projects. The projects can be independent or part of a larger program. Sustainable value is achieved as the enterprise builds on its experiences. Preferably, all BPM projects would have top-level sponsorship, have strong roots in organisational strategy, and a sound process architecture. But not all BPM projects are created equal. BPM projects can be strategic or tactical to varying degrees. Table 2 shows three different approaches. [Jeston & Nelis, 2008, 1:64-66]

Driver	Description
Strategy-driven	Assumes organisation strategy has been defined and documented. Based on this it has decided to implement a BPM program/project. The approach is top-down.
Issue-driven	Based on operational or business issues within an enterprise f. ex. at department or line of business level. Drivers are typically business opportunities, problems or regulatory issues). More tactical in nature than strategy-driven BPM
Process-driven	A business unit is investigated for process improvement opportunities. Contains a high-level and a subsequent detail process examination. Can be strategic or tactical.

Table 2: BPM Initiation Drivers - [Own production]

To quickly sum up this part of the chapter; BPM can be seen as an initiative that contributes to the achievement of an organisations objectives through improvement, management and control of business processes. The basic tenet is that processes should be managed across their entire life-cycle.

3.3 Service-oriented Architecture

The term *Service-oriented architecture* (SOA) is a likely candidate to being the most ambiguous term used in IT today. The exact meaning often remains a source of confusion, even among scholars and practitioners. Martin Fowler describes how many words or phrases in IT goes through a process of semantic diffusion, akin a kind of “*chinese whispers*”,²⁷ in which the original meaning of the word eventually gets lost. But Fowler also claims that SOA never had a real meaning to begin with. [Fowler, 2006] It does however seem, that the SOA term was coined by Gartner in 1996.²⁸ In short, SOA can be considered an architectural paradigm that emphasises loosely coupled, autonomous, and re-usable services.

3.3.1 What is Service Oriented Architecture?

Definitions of SOA are often divergent, and at times even conflicting. The most common source of divergence seems to be the whether SOA should be considered an abstract architectural style or whether SOA should be considered a concrete technical style. The technical SOA (or Web Service SOA) often refers to a SOA that adheres to specific Web Service standards, such as those originating from W3C, OASIS and WS-I.²⁹ This is the SOA that Thomas Erl refers to as the “*false SOA*”. [Erl, 2005:2] In this thesis, the Web Services SOA will however be considered a tactical implementation of SOA, i.e. Web Services are an open-standard based approach to building SOA.

²⁷ Chinese whispers is “*a game in which each successive participant secretly whispers to the next a phrase or sentence whispered to them by the preceding participant. Cumulative errors from mishearing often result in the sentence heard by the last player differing greatly and amusingly from the one uttered by the first.*” [Wikipedia]

²⁸ Unfortunately I have had no access to the original Gartner article: SSA Research Note SPA-401-068, 12 April 1996

²⁹ The standards are sometimes referred to as the WS-* or even WS-Deathstar standards, due to their complexity

In response to the many conflicting understandings, standards-organisation OASIS has created a SOA reference model (SOA-RM). The SOA-RM is an *“abstract framework for understanding significant entities and relationships between them within a service-oriented environment”*. [SOA-RM:1] With this model, OASIS hopes to spur growth of SOA, as well as to help establish a more common understanding of SOA. Since the model is abstract, concrete architectural models should be derived from the reference model. The high level of abstraction is a main strength of the model; it is not tied to any particular vendor, technology nor technology standard.

According to the reference model, the basic tenet of SOA is *“the task or business function – getting something done”*. [SOA-RM:10] In the reference model, SOA is defined as: [SOA-RM:8]

“a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains “

Service capabilities *“represents a specific function of a service through which the service can be invoked”*. [Erl, 2008:115] So capabilities are essentially the ability to perform a task or an assignment. Capabilities should ideally be created on the basis of needs, but there is not necessarily a one-to-one relationship between needs and capabilities. Sometimes several capabilities must be brought together to satisfy one need, and sometimes a single capability may be able to satisfy more than one need. [SOA-RM:8] Furthermore, invoking a service means the realisation of real-world effects, which includes: [SOA-RM:12]

- information returned in response to a request for that information
- a change to the shared state of defined entities
- or a combination of both.

3.3.2 What are Services?

As the name suggests, services are the central unit of work in SOA. The idea in SOA is that “*services are the mechanism by which needs and capabilities are brought together*” [SOA-RM:9]. This is an important concept; because capabilities and needs can arise independently, services can help align needs and capabilities.

But what exactly is a service? In abstract terms, something can be considered a service, whenever an entity carries out a distinct task in support of someone else. Services can be provided by many different types of entities, such as persons, groups of persons acting collectively, IT-systems or by enterprises themselves. To classify a distinct task as a service, it should be well-defined and relatively isolated from other tasks [Erl, 2008: 68]. In daily life, we are all both providers and consumers of services. Services can be either atomic or composite. An atomic service would f. ex. be a haircut service, where the real-world effect is a haircut, which is obtained in exchange for a payment. Composite services are composed by a range of smaller services, each providing specific and distinct capabilities. An example of a complex services could be a travel package booking service, including flight tickets, transfer, hotel, and insurance.

A conceptual model of a service in SOA can be seen in figure 6. In a very simple sense, services are collections of capabilities. Services can be simple, i.e. encapsulating a single capability or be complex services, encapsulating a collection of capabilities. Capabilities are grouped together in services, usually because they are related by some functional context [Erl: 2008:70], which also improves cohesion.³⁰

³⁰ Cohesion is an informal measure of how functionally related the operations of a software elements are [Larman:290]

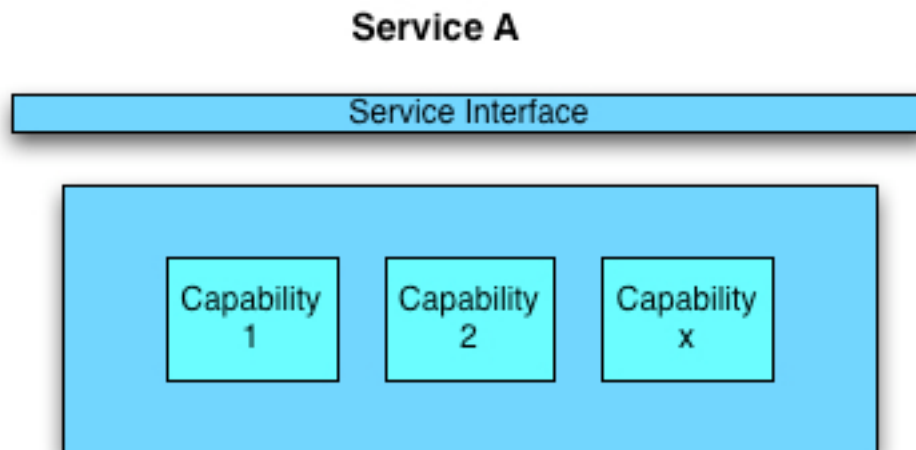


Figure 6: Conceptual Service Model [Own production]

The purpose of accessing the capabilities through a well-defined interface (rather than directly) is to make the implementation details opaque to the service consumer. This provides abstraction between the external and the internal behaviour of the service, freeing the consumer from considering implementation details of the service.

An important concept in SOA is *granularity*, which is a relative measure of the size of the service. In general, services encapsulating a rich set of functionality are considered coarse-grained, whereas services encapsulating a smaller set of functionality are considered fine-grained. Things are however a little more subtle than that, as there are several different types of granularity in relation to services.

There are at least four documented types of granularity in relation to SOA: [Erl, 2008:115-117]

- **Service Granularity:** Refers to the functional scope of the service itself. The overall granularity is not determined by the actual logic that a service encapsulates, but rather the potential logic that the service could encapsulate, within its functional context. An order service would f. ex. be more coarse-grained than an order entry service, even if both were encapsulating the same set of capabilities.
- **Capability Granularity:** The functional scope of a specific capability. Coarse-grained services usually performs more work than fine-grained services.
- **Data Granularity:** The quantity of data a capability needs to exchange in order to carry out its function.
- **Constraint Granularity:** The amount of detail with which a particular constraint on the service is being expressed with.

Determining the right mix of granularities in service design is difficult. A coarse-grained service could f. ex. expose fine-grained capabilities, and a fine-grained service could expose a (relatively) coarse-grained capability. The right granularity depends heavily on context.

While services may follow the same basic pattern, not all services are created equal. In fact, there are different types of services, each with different purposes and characteristics. Several authors suggests taxonomies or classification schemes for service types, each relying on different dimensions or principles for the classification. Different taxonomies are provided in [Lublinsky & Tyomkin], [Sundblad & Sundblad], [Rossberg & Redler] & [Erl, 2008] A common idea in service classification is to classify services according to a hierarchy of volatility; by arranging services into such a hierarchy, it is possible to compose volatile services from less volatile services. This provides isolation of changes.

The most important service type in relation to business processes is the business service. It should be noted, that the term business service is used quite differently in other contexts.³¹ Because business services encapsulates actual business functionality, they will be especially important to align with business requirements. [Lublinsky & Tyomkin:56] Business services should be less volatile than the business processes that they are being consumed by. Another type of service is the entity service. Entity services supplies and protects the data entities that the enterprise depends upon.³² [Rossberg & Redler:285] Entity services are foundational services, which should be stable over time. Finally, the last type be mentioned here is the *utility services*, which are process-agnostic, as they do not carry any business logic. [Erl, 2008:269] A utility service could f. ex. be a generic logging service. From this description it also appears quite evident, that the type of service to a large degree is a determinant of the service granularity. Business services are f. ex. likely to be much more coarse-grained than entity services.

Having seen SOA from the perspective of the individual service, it is now time to proceed the see how service interaction takes place. The service interaction model is another characteristic of the service-oriented architectural style. Figure 7 illustrates a conceptual SOA interaction model,

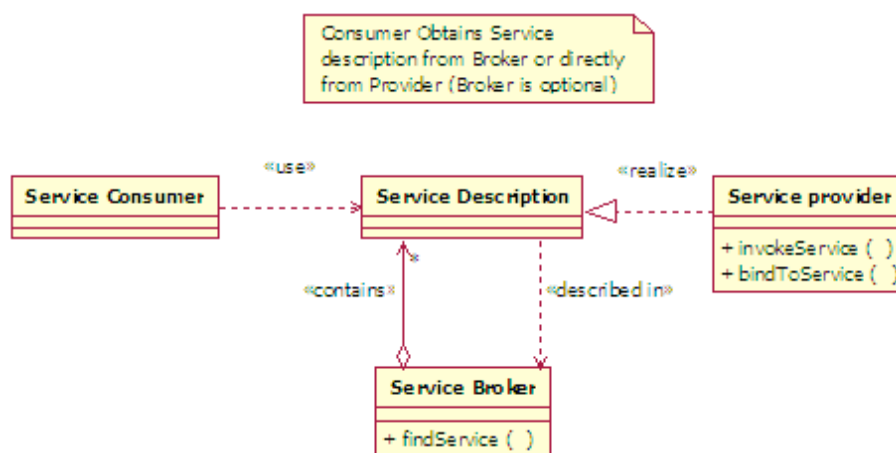


Figure 7: Service Interaction Model [Arsanjani]

³¹ TOGAF does for example use the term in a different way.

³² Such as business objects, i.e. Customer, Employee, Sales-order etc.

The service interaction model separates service interactions into three distinct parties: [Arsanjani]

- **Service provider:** Delivers services to the service consumers. Publishes Service Description and provides implementation of the service. Service consumers may not be known to the provider.
- **Service consumer:** A consumer of the service, who is either aware of the service directly or looks up the service in an appropriate registry
- **Service broker:** Provides and maintains the service registry. The Service Broker is optional but recommended, as it provides for a more loosely coupled interaction style

The service provider and the service consumer are sometimes jointly referred to as service participants. [SOA-RM:9] The service description is not a party to the interaction, but by providing a description of the service, separation of concerns is further promoted between provider and consumer. Effectively, it is possible to achieve separation between “*description, implementation, and binding*” [Arsanjani, 2004]

3.3.3 Loose Coupling

Based on the description of services and the description of the provider/consumer interaction pattern, it is now time to go further into the principle of loose coupling. Coupling refers to the dependency two parts holds on each other. [Erl, 2008:165] If two parts are very dependent on each other, it will be difficult for either of the parts to change behaviour. Therefore, loosely coupled systems generally have the advantage, that they are much easier to change than tightly coupled systems. Loose coupling is for a reason, perhaps the most often mentioned benefit of SOA. Dependency is a measure of coupling and dependency can be divided into: [W3C, 2004, 1]

- **Real dependency:** “*the set of features or services that a system consumes from other systems. The real dependency always exists and cannot be reduced*”
- **Artificial dependency:** “*the set of factors that a system has to comply with in order to consume the features or services provided by other systems*”

One can never expect to eliminate all dependency between two communicating systems. Rather, there will always be a residual (or real) dependency. So the goal is to reduce or eliminate artificial dependency, as well as reducing the cost of the real dependency. The principle of loose coupling is complex, and manifests itself on many different levels. Table 4 shows just how different tightly coupled systems are from the loosely coupled ones.

	Tight Coupled	Loosely Coupled
Interaction	Synchronous	Asynchronous
Messaging Style	RPC	Document
Message Paths	Hard Coded	Routed
Technology Mix	Homogenous	Heterogeneous
Data Types	Dependent	Independent
Syntactic Definition	By Convention	Published Schema
Bindings	Fixed and early	Delayed
Semantic Adaption	By Re-coding	Via transformation
Software Objective	Re-use, Efficiency	Broad Applicability
Consequences	Anticipated	Unexpected

Table 3: Tight versus Loosely Coupled Systems [Kaye:133]

Loose coupling is of course not a panacea. The extra cost of developing loosely coupled systems may in itself deter enterprises from pursuing loose coupling. Moreover, loose coupling should be regarded an ideal that needs to be weighed against other non-functional requirements. Non-functional requirements are *“sometimes known as constraints or quality requirements.”* [Swebok:2-2] ISO 9126-1 is a standard for measuring software quality and is considered: *“One of the most, if not the most, widespread quality standard available in the software engineering community”* [Carvallo & Franch:10]. The ISO standards details the following five main characteristics of non-functional requirements: reliability, usability, efficiency, maintainability & portability. So we should always expect to make trade-offs between the ideal of loose coupling versus costs and non-functional requirements.

This also means that SOA is not the right solution in all situations. Especially, in some environments, SOA might be rendered inappropriate due to certain non-functional requirements. An obvious example is environments with real time requirements, because SOA typically relies on loosely coupled asynchronous communication. Heavy batch processing may also be difficult to fit into SOA.

The final row of the table shows the most dramatic effect of switching towards more loosely coupled systems; consequences change from from being anticipated to becoming unexpected. Striving towards autonomous and re-usable services improves flexibility, but also makes it more difficult to anticipate future use cases for a service. Thus, it also becomes more difficult to calculate the ROI for the service development. This is likely to pose a challenge to existing models of funding IT-development projects.

3.3.4 The Effects of Service-Oriented Architecture

So far, a lot of ground have been covered in regards to the characteristics of SOA. But in the end, SOA is of little interest, unless it provides benefits to the enterprise. The meaning of abstraction and loose coupling has already been seen on micro-level, i.e. between consumer and provider. But the SOA paradigm has much wider implications. Figure 8 views SOA in a broader perspective. The business processes resides at the upper layer. At the lowest layer, the legacy systems are depicted. The services are located as an intermediary layer between business processes and the legacy systems.³³ Services thus provides acts as an abstraction layer between business and technology. In this context, it may be worthwhile to remember, that activities in business processes were defined as collections of tasks, whereas services were defined as collections of capabilities. There is a big conceptual overlap between activities and services.

³³ For the sake of simplicity only business services are depicted, and services are matched one-to-one with activities. This is of course not a realistic scenario.

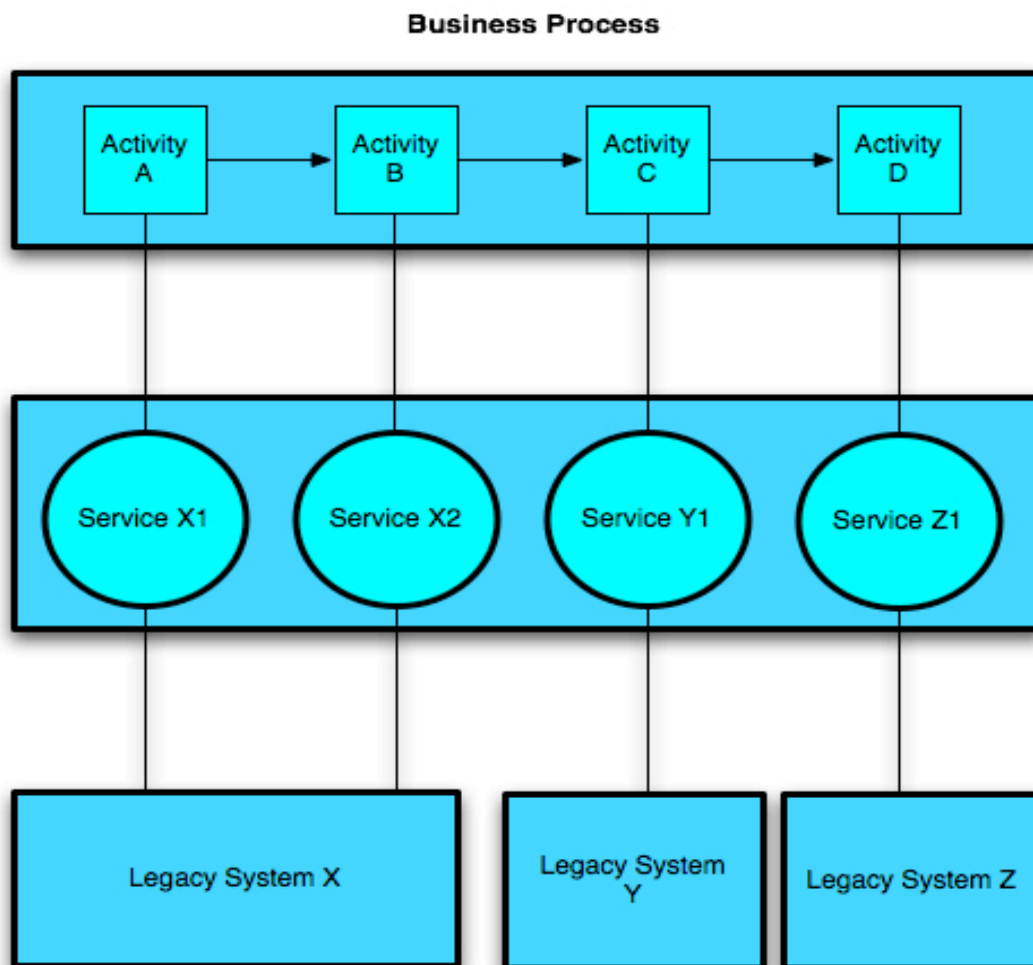


Figure 8: Abstraction of Business Processes [Own production]

A traditional problem with legacy systems is that they turn business logic and implementation details into functional silos. Functional silos are, “*structure[s] that separates its content from outside influences*”. [Bloomberg & Schmelzer:70] From an information systems perspective, a silo is a monolithic structure, where the “*functional logic of a system, has not been properly decomposed and clearly separated from the associated implementation details*”. [Guttman & Matthews] By using services as an abstraction layer, consisting of loosely coupled and autonomous services, the organising logic is separated from the underlying implementation details. Business processes can now be managed independently from the underlying legacy systems.

So SOA does not only provide loose coupling within the IT-architecture, but also looser coupling between the business and the IT-architecture. Business processes can thus easier be composed by the business. This allows for much more flexible architectures.

3.4 Chapter Summary

The purpose of this chapter was to describe EA, BPM and SOA as individual entities. This would lay the foundation for the next chapters.

I sat out describing EA as an enterprise-wide and holistic approach to building architectures. EA emphasises documenting the enterprise in the *as-is* state and in a desired *to-be* state. A main goal of EA was to escape the bottom-up thinking about resource development in the enterprise, thus providing alignment.

Two dominant EA frameworks were then described. The Zachman Framework were seen as the archetypical taxonomic framework, as it mainly serves as a structure for classifying artefacts, and for defining the scope of the EA artefacts to be produced. TOGAF in contrast, was seen as a more methodology-oriented approach, focused on describing the types of deliverables to produce and on how to produce them.

I then proceeded to defining and describing BPM, which was seen as an initiative that seeks to contribute to the achievement of an organisations objectives. Business processes are sequences of activities, that should ideally contribute to fulfil a goal or a policy of the enterprise. A central tenet is that business processes are having a life-cycle, which needs to be managed across all phases. A particular important topic concerns the digitisation of business processes. Digitisation implies automation, but it can be seen on two different levels. The first level concerns automation of business processes, sub-processes or activities. The second level concerns the digitisation of the business process management process itself. This would included using digitised models to close the gap between models and execution. A BPMS was found to be an important tool in this context.

Finally, I went on to describe SOA as an architectural paradigm. The technical SOA was regarded as tactical implementation of the SOA paradigm. The central unit of work in SOA are services, which are collections of capabilities. An important SOA issue is the question of granularity. Service granularity can be seen as a relative measure of the size of a component, usually referring to the functionality of a service. But as there are different types of granularity, defining the right granularity would depend on context.

The idea of different types of services, classified according to a hierarchy of volatility was also described. Such a hierarchy would allow more volatile services to be composed from less volatile services.

Loose coupling was seen as an ideal, that would provide for more flexible architectures. But it was also found, that loose coupling had to be weighed against other non-functional requirements. Finally, I looked at the impact of the SOA paradigm on a scale greater than that of the relationship between the individual consumer and provider. It was found, that SOA could help separating the organising logic of business processes from the underlying implementation details. This is likely to provide a much more flexible way to compose business processes.

Module 2 – The Sum of the Parts

Chapter 4 – BPM and SOA

The purpose of this chapter is to analyse how BPM and SOA fits together, and especially to understand how the combination can help improve agility. The term BPM-SOA will be used as a shorthand term referring to BPM and SOA together.³⁴

At surface level, BPM and SOA may seem like two very distinct entities. Table 4 highlights some of the findings from chapter 3.

	<u>BPM</u>	<u>SOA</u>
What	The achievement of an organization's objectives through the improvement, management and control of essential business processes	A paradigm for developing and using capabilities across physical and ownership boundaries
Who	Business-driven	IT-driven
Unit of work	Business processes (sequence of activities)	Services (collections of capabilities)
Scope	Project	Enterprise-wide (potentially)

Table 4: BPM versus SOA [Own production]

³⁴ The term “*Process Driven SOA*” could have been used as well. This would however de-emphasise the use of BPM as a methodology.

Several theories can be raised as to the nature of the relationship between BPM and SOA. The following two dimensions will be used to evaluate the relationship between them:

- **Orthogonal versus Overlapping:** The term *orthogonal* originates from the field of geometry, where orthogonality involves something having right angles. In computer science however, two entities are considered orthogonal if they are completely distinct, isolated and does not produce any side-effects towards each other. In the context of BPM-SOA, orthogonality would mean that BPM and SOA can be managed as two completely independent entities. This would be one extreme end of a continuum. The opposite extreme is a situation where two entities are completely overlapping (i.e. essentially a theory of sameness).
- **Opposite versus Complementary:** The second dimension concerns the value of combining BPM and SOA. The question is whether BPM and SOA are entities that detracts from each other because they are conflicting, or whether they are entities that completes each other, and emphasises each others strengths.

The chapter will progress in this way; I will start by fleshing out a vision of BPM-SOA. This vision will describe the nirvana of BPM-SOA. Based on this vision, I will establish a working hypothesis as to the relationship between BPM and SOA. The hypothesis will describe where BPM-SOA would be expected to be located within the two dimensions. The relationship between BPM and SOA will then be explored through two different views. Each view will contribute to the larger picture, but leave out details irrelevant to the particular view.

The two views are:

- **Model-driven view:** In this view, BPM-SOA will be seen as a model-driven paradigm, which emphasises the use of models rather than code. In this way, it should be possible to turn business process models into executable models. I will analyse whether this is in fact possible or if there are semantic gaps preventing this.
- **Methodology-driven view:** In this view, I will see BPM and SOA not just as individual entities, but also as methodologies that can work together.

At the end of the chapter, I will be able to discuss the relationship between BPM and SOA. In particular, I will raise and answer the question: “Is BPM the Business Case for SOA?”. Based on this, I will proceed to the conclusion, where I will be able to either validate or reject the working hypothesis.

4.1 The Vision of BPM-SOA

The purpose of this section is to give a 50.000 feet view on the relationship between BPM and SOA. The vision here is based on the way the relationship between BPM and SOA is often popularised by vendors, in blog-posts etc. In other words; this section is not for blunt criticism.

The combination of BPM and SOA is often depicted as a stack, consisting of a number of independent layers. This stack-based view of BPM-SOA is illustrated in figure 9. One is likely find BPM-SOA depicted similarly in many vendor presentations or vendor reports. BPM will be situated as the upper layer of the model. The business processes interacts with the underlying technology by consuming a layer of business services. By exposing capabilities through well-defined service interfaces, the business services becomes an abstraction layer between business and technology.

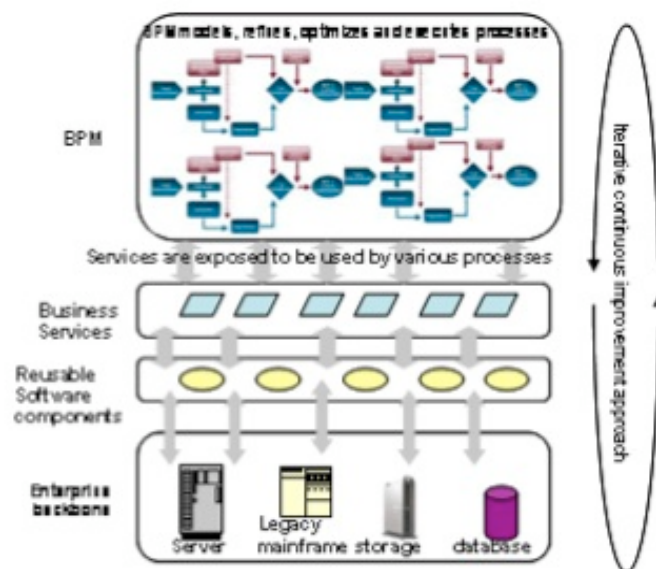


Figure 9: BPM-SOA Stack [Kamoun]

The overall vision of this BPM-SOA stack is to empower the business analyst. The nirvana of BPM-SOA would be a an architecture, where the business analyst is able to manage business processes across the entire life-cycle, with little or no intervention from IT. Looking from a perspective of BPM-SOA, the business process life-cycle would ideally be managed like this:³⁵

- **Analysis:** The business analyst performs analysis of the business process. The analysis can be initiated f. ex. on basis of a process improvement project or because process failures have been detected.
- **Design:** Through the use of user-friendly modelling tools, the business analyst can alter and design business processes. Preferably, the business analyst is also able to simulate the proposed business process design.
- **Implement:** The designed or changed business process is automatically deployed to a run-time execution engine, which is then capable of executing the business process as necessary.
- **Run/Monitoring:** Once the process has been deployed and active, the business analyst is capable of monitoring the business processes in order to investigate performance, and to detect process failures.

It sounds almost like the 3rd wave of BPM, as expressed by Smith & Fingar, who states that: *“the gap between business and IT is not bridged, but rather obliterated”*. [Smith & Fingar] This vision do also seem remarkably similar to the description of the BPMS discussed in chapter 3. So in which ways does this approach differ from the integrated BPMS solution? The main difference is that the traditional integrated BPMS solution would usually be implemented on a fairly closed stack of proprietary technologies, including a proprietary integration platform. The BPM-SOA solution in contrast, emphasises the use of industry standards. The idea is, that the business analyst designs business processes in a standard modelling language. The business process models are then passed down the stack for execution on the standard-based execution engine, which is then capable of executing the SOA services.

³⁵ The life-cycle phases are identical to the phases described in chapter 3.

SOA here serves as the flexible underlying IT-architecture, that qua re-usable and coarse-grained business services can provide a level of integration and flexibility that would otherwise be difficult to achieve. So in this context, SOA is regarded as the plumbing underneath BPM. Based on this vision of the relationship between BPM and SOA, it appears that the two entities are complementary, but also orthogonal. In the rest of the chapter, I will therefore explore the following working hypothesis:

BPM and SOA are orthogonal but complementary entities

4.2 The Model-driven View

In the model-driven view, BPM-SOA will be regarded as a kind of Model-Driven Engineering (MDE). MDE is a form of software development that emphasise production and use of models as primitives, rather than code. As with the object oriented paradigm where everything was considered objects, everything is considered models in MDE. [Bézivin, 2005:1]. Under the model-driven paradigm, models will be important as both design and run-time artefacts. The purpose of using models is to raise the level of *abstraction* above code-level, and to enable separation of concerns³⁶ between specification and implementation.

Since models are central primitives in MDE, understanding the term is vital. The term *model* has many different connotations to people, and is used in many different contexts. People are able to recognise different types of models, such as car models, fashion models or models that are physical replicas. Moreover, models are a well-recognised within the field of *software engineering*. Rothenberg provides a rich description on the use of models:

"Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality." [Rothenberg:1]

So models are considered simplified versions of something that exists in reality. These models can be used as tools for communication, analysis or design. Models are often presented by a combination of drawings and text. The text could be expressed either in natural language or in a custom modelling language. [OMG, 2003:2-2]

³⁶ I.e. modularising or separating distinct concerns into different areas, so that each has a cohesive purpose [Larman:441]

In this context there are two important types of models to consider:³⁷ [Dietz, p. 64]

- **Conceptual model:** An abstraction or a conceptualisation of a concrete (proposed) system. The model is expressed in an informal way, such as via drawings or natural language.
- **Symbolic model:** A symbolic model can also be considered a conceptual model, but differs because it is expressed in a formal language, i.e. by using symbols. A symbolic model can thus be considered a *formulation* of a conceptual model.

MDE is concerned with using symbolic models for representing systems at different levels of abstraction. Model transformations are central to this approach: a model at a given level of abstraction can be used as a source model, being transformed to a target model at another level of abstraction.³⁸ Transformations are achieved “*by using a set of rules, specifying the mapping between source and target model*”. [Brahe & Bordbar:3] The word *transformation* may imply that this is an automatic process (in parts or in whole). This is certainly an ideal, but in some cases, human intervention is required, f. ex. by elaborating on source models before they can be transformed to target models. It is also worth noting, that transformations (which are sometimes also called translations) implies some level of understanding of the meaning of the model. This is in contrast with transcoding, i.e. the conversion of a model from one format to another.³⁹ [Dietz: 65]

³⁷ Other types of models exist, such as concrete models. A concrete model of a system would be considered an imitation

³⁸ This is the most typical scenario. The reverse could also happen

³⁹ Which would just be different syntactical representations of the same model

It is difficult to talk about MDE without also mentioning Model Driven Architecture (MDA). MDA is a framework and a set of standards from the Object Management Group (OMG), which can be regarded as one of the most well-known branches of MDE.⁴⁰ Recently, OMG has also extended focus into the business domain. [Lankhorst et al.:27] At this point, MDA does however not play a significant role within the BPM-SOA space. An important reason for this is, that none of the most important languages for BPM-SOA has yet become MDA-based. But I have decided to include a description of MDA anyway for two important reasons. Firstly, a basic description of MDA can be used to demonstrate important model-driven principles. Secondly, OMG is trying to position MDA within the BPM-SOA space. Thus, MDA has the potential to become an important future player in this space. I will therefore return to the subject of MDA several times in latter parts of this chapter.

The basic structure of MDA can be described through three different viewpoints, which operates at different layers of abstraction. The three viewpoints separates business modelling from the underlying implementation details:

- **Computation-Independent Model (CIM):** The CIM model is at the highest level of abstraction, and contains no technology details. This is where the business modelling (or domain modelling) resides. The business analyst is the primary stakeholders.
- **Platform-Independent Model (PIM):** The PIM model, is the first level that describes the system. Although PIM models contains computational logic, they still do not contain implementation details. The primary stakeholder is the solution architect.
- **Platform-Specific Model (PSM):** At the lowest level of abstraction the PSM models are found. This is the level of abstraction just above the code itself. Typical examples of PSM models are models of object oriented classes, such as in UML Class Diagrams. PSM models are aimed at the system developer.

40 Another popular MDE framework is the Eclipse Modeling framework [EMF]

The process of turning a high-level business model into code would follow this pattern: Initially, business modelling would be performed at the highest level of abstraction (CIM level), without any attention being paid to implementation details. Through a series of steps, the CIM model is transformed into a system model, that contains computational logic, but no implementation details (PIM). The PIM can then be turned into a model that is specific to the underlying platform. Finally, the platform specific model (PSM) can be transformed into executable code. This process enables business models to become not only important artefacts for analysis and design, but also important run-time artefacts.

BPM-SOA can be regarded as model-driven, because we are using models to “*direct the course of understanding, design, construction, deployment, operation, maintenance and modification*”. [OMG, 2003:2-2]. Both the BPM-SOA envisioned earlier and MDE favours:

- The use of symbolic models as central primitives, rather than code
- Business modelling is performed at a high level of abstraction
- High level models are turned into lower level models through transformations
- Eventually, models are transformed into executable code

But to realise the vision of BPM-SOA, it should be possible to establish a complete modelling value chain from business modelling and down to deployment. Any semantic gaps in this value-chain, would become an impediment to continuous improvements. [Khoshafian, 2005]

4.2.1 Semantic Gaps in BPM-SOA

The main purpose of this section is to investigate, whether such a modelling value-chain can in fact be established without semantic gaps.

This part of the chapter will fall in three main parts. First, I will propose a modelling value-chain, based on the most likely candidates for modelling languages that would fit each layer in the value-chain. Secondly, I will describe some important elements to defining modelling languages. Understanding these elements, will be required for understanding the issues that can crop up, when transforming between two modelling languages. Lastly, I will analyse the proposed value-chain for semantic gaps.

4.2.1.1 The Modelling Value-chain for BPM-SOA

In the following a modelling value-chain for BPM-SOA will be proposed. Figure 10 depicts the conceptual value-chain, including an overview on how the different layers fits within the three MDA perspectives. The different layers are described below the figure, including the most likely candidates for modelling languages fitting the purpose of each individual layer.

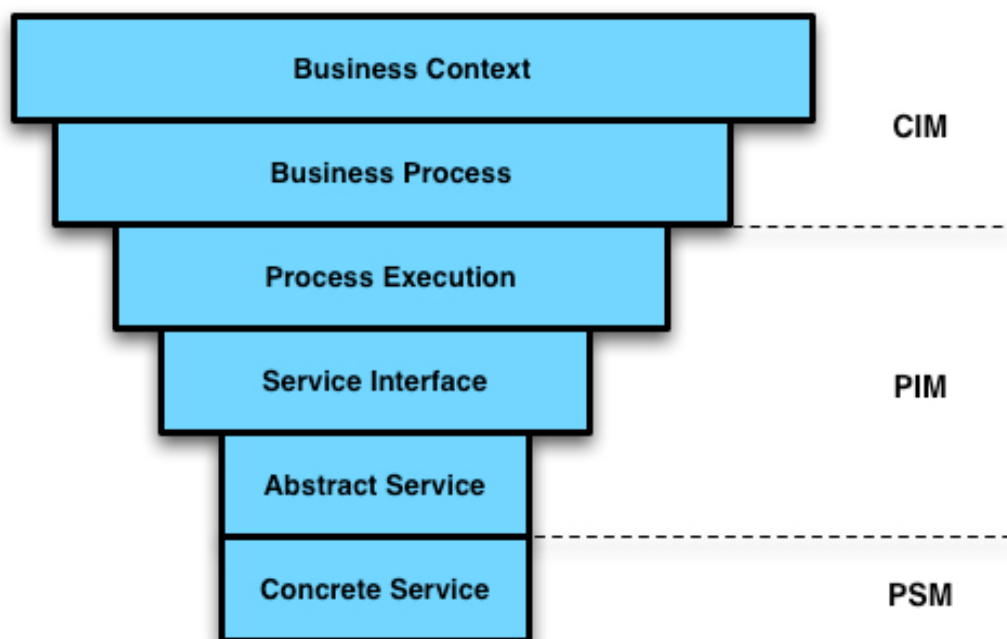


Figure 10: Modelling Value-chain [Own production]

The **Business Context** concerns modelling of the context and the dependencies which surrounds processes and services. As such, the business context is setting the overall frame that BPM-SOA projects operates within. Some of the contextual business information captured here would be strategy and vision, business policies, decision rules, and organisational models. [Khoshafian, 2005:105-106] Moreover, goals and objectives are important process-related meta-data. An important thing to note is that although recent years has seen a flux of new business-related modelling languages,⁴¹ not all aspects of the business context can yet be expected to be captured in standardised modelling languages. Much contextual information will still be found in natural language documents, such as strategy papers.

41 For example the Business Motivation Model [BMM], the Semantics of Business Vocabulary and Business Rules [SBVR] and the Organizational Structure Metamodel [OSM]

The second layer concerns the **Business Process** Modelling, which of course is central to BPM-SOA. Previously, business process modellers had to rely on proprietary notation standards for drawing up business process diagrams. Today, business process diagrams are usually modelled with the Business Process Modelling Notation 1.1 (BPMN) standard, which is emerging as the de facto standard. BPMN is primarily a standardised graphical notation for business process modelling, rather than a standard for creating executable process models.

The next layer is the **Process Execution Modelling**. In this layer, attention is turned away from analysis and design, and towards the execution of business processes. The WS-BPEL 2.0 standard (hereafter just referred to as BPEL) is “*an XML based-language for describing business processes and business interaction protocols*”. [Brahe & Bordbar:3] BPEL provides the facilities for combining and co-ordinating service invocations. [Brahe & Bordbar:3] In this sense, the BPEL language can be considered a process integration model.

The third layer defines the **Service Interface Modelling**. During the investigation of SOA in chapter 3, the importance of using a service interface to hide implementation details was emphasised. Two main standards are used to define the technical service interface:

- **Web Services Description Language (WSDL):** WSDL is an XML format for describing Web Services and how to access them.
- **XML Schema Definition (XSD):** XSD is used for describing the structures and for constraining the contents of XML documents. WSDL relies on XML Schema Definition (XSD) for defining input- and output data types.⁴²

42 Moreover, BPEL also relies on XSD

The two final layers illustrates service modelling. At the time of writing this thesis, there are no generally accepted way to model services, but some proposals have been put forward. See f. ex. [Emig et al.] and [UMPS] In [Emig et al.] service modelling is suggested being divided into an abstract part, and a concrete part. The abstract service model would be used to model services in a platform independent way, i.e. detached from the run-time environment. In the case of composite services, the abstract model will also describe the relationship to other services. [Emig et al.:1] The concrete service model in contrast, includes deployment information. At run-time several instances of the same abstract service may exist. The concrete service model is derived from the abstract service model, with the addition of deployment specific information, such as binding type and service endpoint references. [Emig et al.:3] This is completely in line with the WSDL standard which states that: *“The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint”*. [WSDL]

4.2.1.2 Elements of Domain Specific Modelling Languages

Before proceeding, it will be necessary to give an introduction to domain specific modelling languages. This introduction will serve as basis for understanding some of the issues that can arise when attempting to bridge two modelling spaces. The introduction will be fairly thorough, as a sound understanding of modelling languages will be vital knowledge in the rest of this thesis.

The notion of transforming symbolic model have been stressed throughout this chapter. The definition of symbolic models given earlier, stating that symbolic models are “expressed in a formal language” is however not particular precise. In this section a more thorough description of language formalism will be given. It will be the point, that there are certain elements that are central to defining (or describing) languages formally. The language formalism given here, applies to software languages as a whole, including programming and modelling languages. But in this context, Domain Specific Modelling Languages (DSML's) are of particular interest. A DSML addresses a particular problem domain, encapsulating domain-specific knowledge using a domain specific terminology [Erche, Wagner & Hein:1037] Such languages are vital elements in the modelling value-chain.

A simple language definition is:

“A language L is the set of all linguistic utterances of L .” [Kleppe:2]

The term *linguistic utterances* refers to the expressions that can be given in a certain language. The definition implies, that a language description should contain syntactical rules.

Syntax can however be divided into two types: [Kleppe:3] [Erche, Wagner & Hein:1037]

- **Abstract Syntax:** The abstract syntax encapsulates language concepts, by defining elements, their relationships and their constraints. Meta-models can be used as formalism to express the abstract syntax.
- **Concrete Syntax:** The meta-model does however not describe how the language is presented to the user [Kleppe:3] This is the role of the concrete syntax, which f. ex. can be a graphical format, a textual format or a file format.⁴³

Abstract and concrete syntax are mapped towards each other through syntactic mapping. It is important, that the abstract and the concrete syntax are not the same. Rather, the abstract syntax should be considered the “backbone” of a modelling language. It is f. ex. often desirable to have several concrete representations of the same language; one concrete syntax would f. ex. define the graphical notation, whereas another concrete syntax would define the serialisation format. Finally, different concrete syntax's of the same language, should be syntactically mapped to the same abstract syntax. If not, they would essentially be different languages.

A more thorough definition of the *language* term provides further insights:

“A language description of language L is the set of rules according to which the linguistic utterances of L are structured, optionally combined with a description of the intended meaning of the linguistic utterances” [Kleppe:2]

This definition also includes optional semantics. It could however be argued, that semantics ought to be mandatory, as it provides the basis for a common understanding of the language. Two additional elements are used to define semantics: [Erche, Wagner & Hein:1037]

- **Semantic Domain:** The meaning of the abstract syntax elements is defined by the semantic domain.
- **Semantic Mapping:** Elements of both the semantic domain and the abstract domain are mapped

⁴³ Abstract syntax is also often referred to as the notation

All semantics are however not created equal. To be able to execute a model on a technical platform, the execution semantics needs to be defined very precisely. There is no exact way to discern whether a semantic description can be considered a description of execution semantics. But in general, execution semantics would provide a step-by-step description on how to execute any given abstract syntax element on a run-time platform.⁴⁴

Finally, a graphic illustration of language descriptions elements can be seen in figure 11.

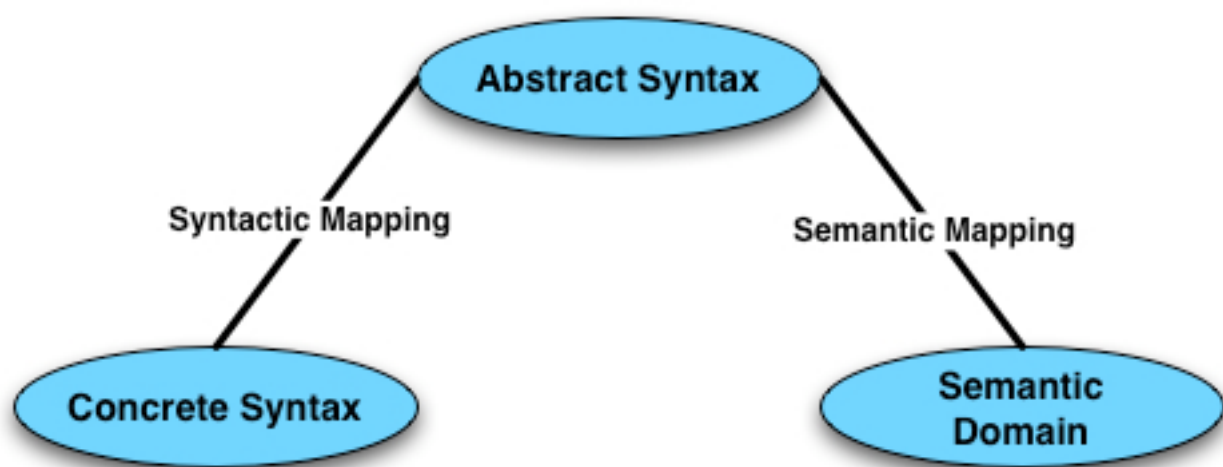


Figure 11: Language Description Elements [Own production]

There are important efforts underway in order to standardise some of the elements. Especially, the OMG is active in this space, attempting to establish important standards around MDA. The Meta-object Facility (MOF) is an OMG standard that defines a standard way to express meta-models. The MOF language is a minimal set of constructs that can be used to model other modelling languages. [Gasevic, Djuric & Devedzic:114] As such, it can be considered a meta-meta-language for expressing meta-models. By establishing MOF as a single language for the specification of meta-models, MOF becomes a bridge that ensures a modest degree of commonalities between different languages.

⁴⁴ Execution semantics is also sometimes referred to as operational semantics or process semantics.

Moreover, OMG has created the XML Meta-data Interchange (XMI) format. XMI is a standard for serialising MOF meta-models and MOF based models to XML. Because XMI also allows for the creation of XML schemas for MOF based models, it can implicitly be used to define a concrete syntax for modelling languages. There are however some interoperability issues with XMI. Appendix F provides a more thorough description of MDA and MOF, as well as a discussion of the interoperability issues with XMI.

As shall be seen later, there are efforts underway to use MDA, MOF and XMI for BPM and SOA.

4.2.1.3 Semantic Gap – From Process Modelling to Process Execution

It is now time to look at the potential gap between the process modelling (BPMN) and the process execution (BPEL) spaces. Crossing this chasm is challenging in part because BPMN itself has some limitations, and in part because mapping from BPMN to BPEL is a complex endeavour.

As for the BPMN standard itself, it is primarily a notation for drawing business process diagrams. It does not attempt to define execution semantics, and there is no official meta-model attached either.⁴⁵ Furthermore, there is no standardised approach to serialising BPMN models to XML. Proprietary meta-models and serialisation formats have been derived from the BPMN specification, but the lack of standardisation does hamper interoperability.

⁴⁵ According to [Silver, 2006] a non-public draft of a BPMN 1.1 meta-model do exist

All this should not be a problem, as the BPMN chapter contains a chapter on how to map to BPEL. Things are however more complex than so. As the BPMN FAQ states: *“By design there are some limitations on the process topologies that can be described in BPEL, so it is possible to represent processes in BPMN that cannot be mapped to BPEL”*. [BPMN, 2005] The two languages has origins in different backgrounds and supports different phases in the business process life-cycle. This has led to conceptual mismatches between BPMN and BPEL. [Recker & Mendling] One key problem is the divergence in the level of expressiveness provided by the two languages; BPMN provides a much richer set of modelling constructs than BPEL [Recker & Mendling]. BPEL does f. ex. have shortcomings in regards to managing human tasks and sub-processes [Julic & Kant:209] So mapping from BPMN to BPEL, either requires using a basic set of BPMN modelling constructs, risking making models that cannot be transformed to BPEL without information loss or defining additional extensions to BPEL to accommodate these shortcomings.⁴⁶

Furthermore, BPMN and BPEL are not aligned particular well in regards to their underlying structure. The graph⁴⁷ nature of BPMN does not match well with the more block-oriented⁴⁸ approach to BPEL, which further impedes the opportunities for making standard mappings between BPMN and BPEL. [Joergensen]. Finally, because BPEL does not contain a graphical notation, there is not way to preserve the diagram layout when performing round-tripping.

So the relationship between BPMN and BPEL is far from optimal; there is no standard way of mapping between BPMN and BPEL, and BPMN is clearly not itself adequate as an execution language.

⁴⁶ BPMN and BPEL extensions has f. ex. been added to the Oracle BPA and the BPEL Process Manager products [Julic & Kant:209]

⁴⁷ The graph theoretic approach holds that from a particular node you can only move to another that is connected [Harrison-Broninski:124]

⁴⁸ Block-structured programming takes the approach that groups activities into sequences – even which the can be executed in parallel the idea is to follow one specific activity with one other activity

Towards BPMN 2.0

There are emerging solutions to some of these problems, as there has been ongoing work on creating a new Business Process Meta-model Definition (BPDM) standard. This standard includes a meta-model and a schema for serialising BPMN to XML. The BPDM is a MDA/MOF standard. The BPDM is to be aligned with BPMN under the name of BPMN 2.0 by establishing:

A single specification, entitled Business Process Model and Notation (BPMN 2.0), that defines the notation, metamodel and interchange format, with a modified name that preserves the “BPMN” brand. [BPMN, 2006:1]

Since the BPMN 2.0 is going to be syntactically mapped to the BPDM, it would implicitly be aligned with the semantic domain of the BPDM. In other words; BPMN 2.0 would implicitly have execution semantics defined.

The decision to align the BPMN closer to MDA has however caused quite a debacle. According to [Silver, 2008] a different proposal was submitted by IBM, SAP, Oracle & BEA.⁴⁹ Their proposal “*looks a lot like today’s BPMN, but with a bit of cleanup in the semantics, an explicit metamodel and XML schema*”. [Silver, 2008] The core idea behind their proposal is to make the implicit execution semantics in BPMN 1.1 explicit, but without modifying the semantics. They argue that this is the least disruptive solution for existing adopters of BPMN.⁵⁰ Again with [Silver, 2008] as the source, the outcome of the standardisation process does not seem to be preordained. The standard is currently pending approval. In any case, both of these proposals defines execution semantics for BPMN 2.0, whether the BPDM solution or the “simple” solution wins.

⁴⁹ It should be noted that all the parties in the proposal are BPMS vendors. This is likely to have influenced their interest in the outcome of the BPMN 2.0 work.

⁵⁰ And this is unquestionably the right conclusion to draw. Aligning BPMN 2.0 with MDA would increase the learning curve for current adopters. Whether the solution is better, obviously depends on context.

XML Process Definition Language (XPDL)

One way of eliminating the transformation gap altogether is to create a unified modelling and execution language. An example of such language could be the XML Process Definition Language (XPDL). XPDL is foremost designed as a common interchange format, that allows for persistence and exchange of BPMN process diagrams. This has been achieved by the creation of an extended meta-model, which unifies XPDL and BPMN constructs. As such, XPDL should be completely orthogonal to BPEL; XPDL is a process diagram persistence format, whereas BPEL is a process model execution format. In theory, this would lead to a value chain consisting of BPMN-XPDL-BPEL. [Palmer:54]

XPDL holds a number of important advantages over BPEL. Firstly, being a storage format for BPMN, the two standards are already somewhat better aligned than the uneven levels of expression between BPMN and BPEL. Secondly, because of this alignment, XPDL supports round-tripping without information loss, including preservation of diagram layouts. Finally, XPDL also allows for extensions. These extensions can also be preserved during round-tripping or tool exchange.⁵¹ Things are however a little more subtle than just seeing XPDL as a persistence format. Some proponents are positioning XPDL as a run-time language as well. The main idea is that if business process models are captured in BPMN, then business process execution engines should be able to consume the XPDL files. Some tools are already supporting this today.

But for all the advantages of XPDL, it does not guarantee execution semantics. [Swenson:3] Execution of a process model persisted to XPDL, thus requires external interpretation of execution semantics by the consumer.

⁵¹ Which on the other hand allows for proprietary extensions muddling the standard.

Process Language Feature Comparison

So what is the right way to bridge the gap between process modelling and process execution? Table 5 sums up the features of the four process language variants, and it is quite evident that there is no optimal solution yet.

Standard	Graphical Notation	Execution Semantics	Serialisation Format
BPMN 1.1	Yes	No	No
<i>BPMN 2.0</i> ⁵²	Yes	Yes ⁵³	Yes ⁵⁴
BPEL	No	Yes	Yes
XPDL 2.0	Yes ⁵⁵	No	Yes

Table 5: Process Languages Feature Comparison [Own production]

A common language for modelling and execution would clearly be preferable. But at least for the foreseeable future, modellers will have to use several DSML's, which needs to be mapped against each other. This will invariably create semantic gaps. Opportunities do however exist for a unified language to emerge. It is very difficult to predict the outcome of this format war, as the outcome of the BPMN 2.0 work is still not known. But it is likely that BPMN will retain its status as a de-facto graphical notation for business process diagrams, whether the preferred version will be version 1.1 or version 2.0. Much however hinges on the outcome of BPMN 2.0 process. If the BPMN 2.0 manages to get unscathed through the standardisation process (and without alienating the user base), then it seems like an obvious candidate for a combined modelling and execution language. If not, XPDL may have a chance at playing a pivotal role, if not only because it already has some momentum.

⁵² Request for Proposal

⁵³ Either provided implicitly by the BPDM or by a separate meta-model with execution semantics defined.

⁵⁴ Either as XMI (BPDM) or as a separate XML serialisation schema

⁵⁵ Mapped to BPMN graphic notation

The discussion about modelling standards leads to another observation: it appears that there is still a fundamental semantic gap between business and IT, i.e. there is a foundational trade-off to make. More formal languages are easier to make executable, but restricts the business modeller. Less formal modelling languages empowers the business modeller, but are more difficult to make executable. This observation can be seen reflected in this statement from Francis McCabe:

“Personally, I think that the issue is that we are trying to have it both ways: have an easily understood execution semantics and allow the business modeller to do whatever and however he/she likes.” [McCabe]

Despite the opportunities for a unified modelling and execution language to emerge, there will still exist a deep-running gap between the business modeller and the IT-side. It is likely, that such unified language will affect the way the business modeller has to work. A model that will be used for execution is likely to require more discipline by the business modeller. This is a potential cultural issues to handle.

But for most parts, it is possible to bridge the gap between business modelling and execution through standard modelling languages. Thus, it is possible to empower the business analyst by facilitating the opportunity to dynamically re-configure business processes through re-usable and discoverable services. At this point however, the solution will not be based on standards mappings and/or without proprietary extensions.

4.2.1.4 From Business Modelling to Service Modelling

The gap between business process modelling and execution could with some caveats be bridged. There is however an important limitation; the underlying assumption is that services already exists, are discoverable and are implemented in a way that is useful to the business process modeller. Given the vastness of the enterprise IT-asset portfolio, most assets are not likely to be service-enabled by default. Any enterprise worth its salt, would service-enable systems and components very selectively. Creating a service is an investment decision and should be treated as such.

This begs the question if services can be derived directly, i.e. if it is possible to automatically go from business modelling to service modelling? Figure 12 shows a simplified value-chain.

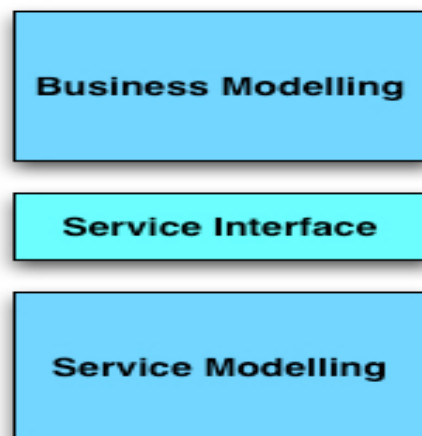


Figure 12: Business versus Service Modelling

The service interface can clearly be created from the business models through a top-down approach. On the other hand, the service interface layer can also be created via a bottom-up approach; by defining the abstract service model, and by enriching the abstract service model with deployment information, it is possible to create a concrete service model. Since this concrete service models now contains both interfaces and deployment information, generating the technical interface, consisting of WSDL and the XSD is possible.

But it is not possible to automatically cross the service interface layer from either direction. Neither BPM, SOA nor the two together, can come up with the right re-usable services. [Guttman] As expected, the service interface acts as an intermediary layer between the two worlds. Human cognition and human work will still be needed in order to close this semantic gap. Especially, collaboration between stake-holders, such as business analysts, solution architects and system developers will be needed.

To return to the discussion about the vision of BPM-SOA, I will quote a blog-post by Michael Guttman, who reflects on a presentation at a large financial institution. Several vendors were presenting their view on BPM and SOA together: [Guttman]

“In general, the vision presented was a brave new world where business analysts can simply compose the new or improved business processes they need from a set of reusable business components, after which some run-time execution BPM engine (nowadays usually based on BPEL) will end up invoking the appropriate reusable SOA-based services to execute those processes.

To this end, each of the various presenters happily showed off slideware of his own company's BPM/SOA 'marketecture'”

This view is very similar to the vision that I presented at the beginning of the chapter. Guttman further states:

“As with most such magic tricks, the presenter must get the audience to believe they 'saw' something that didn't really happen. In this case, this involves getting the audience to believe that, just using BPM and SOA, they will automagically come up with an appropriate set of reusable, recomposable components at both the BPM and SOA layers, plus an efficient mapping between the two.” [Guttman]

Although perhaps a bit polemic, the observation by Michael Guttman is true; there is absolutely nothing new in BPM nor in SOA that would warrant any claims as to automated software development. Rather, claiming the opposite would be like trying to brand BPM-SOA as Case Tools 2.0. The combination of BPM-SOA is not a silver-bullet.

This is not to say that a model-driven approach to BPM-SOA can not provide value; it certainly can. It provides the value that is normally associated with MDE. By using models rather than the code, complexity is reduced and this can improve developer productivity. Moreover, being able to automate or semi-automate transformations in some parts of the value-chain, can potentially reduce the manual coding effort. Thus, a model-driven approach can help reduce the *time to market* for new solutions, as well as reducing the *cost of development and integration*.

And once discoverable services has been developed, the model-driven approach actually allows for dynamic composition of business processes. But in addition to the model-driven approach, formal methodologies are needed in order to bridge the semantic gap, including identifying and specifying new services, and for making changes to existing services. In this sense, BPM and SOA can no longer be considered orthogonal. How this semantic gap should be closed, will be the main topic in the methodology-driven perspective.

4.3 The Methodology-driven View

As was seen in the previous part of the chapter, viewing BPM-SOA purely as a technological stack does not hold. Rather, formal methodologies are needed in order to close the semantic gap. In this part of the chapter, BPM and SOA will be seen not just as technologies, but also as methodologies that works together.

4.3.1 Introduction

Just like their business process counterparts, services in SOA are also considered having their own life-cycle, which needs to be managed. Table 6 describes the three main phases of the service life-cycle. [Seeley, 2008]

Life-cycle phase	Description
Design-time	Identification, specification, and realisation of service.
Run-time	At run-time services are invoked by consumers. Policies are further enforced at run-time. ⁵⁶
Change-time	To provide a flexible platform, that can be dynamically re-configured, much of the configuration in SOA does take place in declarative configuration files based on XML. This includes security policies and service contracts. Providing configuration through declarative configuration files, eliminates the need for recompilation at change-time.

Table 6: Life-cycle Phases of Services in SOA [Own production]

The purpose of this section is to understand how business processes interfaces with the service life-cycle. The main emphasis will be on managing the life-cycle of business services, as these are the ones that will be consumed directly by processes. Managing entity services do share some of the same issues as business services.⁵⁷

⁵⁶ In reality, service policies would have their own life-cycle as well, as they are supposed to be managed independently of the service. For the sake of simplicity I have omitted service policies from this analysis.

⁵⁷ Although to a lesser degree, as entity services are supposed to be more fine-grained and less volatile.

The subject of managing all life-cycle phases of a service is a considerable subject. Therefore, only the most pressing issues in bridging the semantic gap will be dealt with here. The analysis will cover the design-time tasks of identifying and specifying services, as well as how to manage the change-time phase.

Service identification includes identifying the right services to build. A key objective will be to devise a way to use BPM to identify candidate services. Service specification in contrast, includes determining the right service granularity in order to balance a number of different requirements.

Finally, as for the change-time phase, the key objective of the analysis will be to find out how coupling affects the ability to change services. It will be an important point, that although services and their consumers are loosely coupled, there will still be residual coupling that must be managed.

4.3.2 Design-time (Service Identification and Service Specification)

A dominant approach to software engineering is the *Object-Oriented Analysis and Design (OOAD)*. Using the words analysis and design has a special meaning. The emphasis of analysis is to “*investigate the problem and requirements, rather than finding a solution*”. [Larman:6] In contrast, the emphasis of design is to create conceptual solutions to the requirements. In combination, the purpose of analysis and design is to understand the problem, the requirements, and to create conceptual solutions to the requirements. OOAD is however concerned with micro-level abstractions, such as objects and classes [Zimmermann, Krogdahl & Gee] Such a paradigm would not be suitable for designing coarse-grained services. Instead, a new paradigm should emerge in the form of *Service-Oriented Analysis and Design (SOAD)*.

A sub-set of SOAD would be to identify and specify the right services to build. An approach to identification, specification and realisation of services can be found in IBM's framework for Service-Oriented Modelling and Analysis (SOMA). SOMA suggests employing a combined approach to service identification, which contains: ⁵⁸ [Arsanjani]

- Existing Assets Analysis (bottom-up)
- Domain Decomposition (top-down)

The SOMA approach highlights a fundamental issue in service design. Even if a top-down approach allows the enterprise to set the context for the service design, such an approach would not provide opportunities for re-use of assets. Few development projects can be seen as “green field” efforts, because enterprises typically have vast portfolios of existing systems, components and services. So a combined top-down and bottom-up approach to service design is needed.

⁵⁸ The method also includes *Goal-Service modelling* which is a middle-out method to identify services not found through domain composition nor existing asset analysis

4.3.2.1 Using Business Process Management for Service Design

The SOMA approach is certainly valuable as a framework for service design, but it lacks a structured method for performing the domain decomposition. BPM can however be seen as a tool for this. There are two main advantages of using BPM for domain decomposition. Firstly, because BPM analysis and design involves decomposing processes into a sequence of activities, important insights are given about the work to be performed performed during the process. Secondly, because BPM is naturally business-oriented, it should be expected that much process-related meta-data will be uncovered or determined during analysis and design. This provides a sound basis for establishing the right requirements for the service design. Figure 13 depicts a conceptual model for decomposing the domain via BPM.

The model does in some ways resemble the modelling value-chain presented earlier in the chapter. It is important to note, that the use of this decomposition process does not preclude the use of modelling. Rather, such an approach would be encouraged. The two perspectives are not mutually exclusives.

The model takes as granted, that domain decomposition within an appropriate scope has already been decided. In this way. business process improvements becomes drivers for service design. Strategy-driven BPM projects would usually start by performing a value chain analysis, until all relevant processes and sub-processes have been identified. If a BPM project is being driven by less strategic needs, such as issue-driven or process-driven improvement projects, then the domain decomposition would more likely be driven by use-cases, and single processes would typically be targeted for analysis and design.



Figure 13: Domain Decomposition [Own production]

The first step in the domain decomposition is to document the business process from a high-level perspective. Obviously, it is vital to gain an understanding of the purpose of the process, as well as to identify the goals or policies that the process should support. Such meta-data, gives direction for the rest of the domain decomposition process.

The second step is concerned with decomposing the process itself. By identifying and documenting the activities in the process, the enterprise is given important insights as to the work being performed during the process. The enterprise should also understand the control flow, including the activities related to the control flow.⁵⁹ Business objects and business documents are also important entities. Business objects are abstract or conceptual representations of things in the business domain. [Jenz:8] Business documents are the set of information components that are interchanged as part of the business activities. [Jenz:8] Business documents are closely related to business objects, as they are the physical counterparts to the abstract business objects. Moreover, the information flow should be documented. The information flow describes the *“relationship between a business activity and a business document or between a business activity and a business object”*. [Jenz:9] Business activities may read, update, or create business documents or business objects.

The third step is aimed at identifying candidate services. By breaking down the individual business processes into a sequence of activities, the enterprise ends up with a number of activities, which could potentially become services. By filtering out activities that for various reasons are not suitable as service candidates, a smaller set of candidate services is identified. Some activities are not likely service candidates, f. ex. because the activities are clearly human task oriented, such as calling a customer on the phone. Another likely selection criteria is cost versus benefit. Candidate services will be selected based on an initial cost justification. There is no need in spending resources on specifying services that can not be expected to have a positive ROI.

⁵⁹ The control flow is also vital for establishing the process execution model.

The filtering process can be demonstrated with a simple example. Figure 14 shows a sample business process, which is a simple order entry process. A customer order is checked against the customer credit history. If the order is approved, then the order is shipped to the customer. If the order is rejected, the customer is being contacted. The “*Ship Order*” and the “*Contact Customer*” activities are not likely service candidates; their names indicates that these tasks are primarily human oriented. The *Analyse Customer Order* and *Check Credit History* are however are likely service candidates.

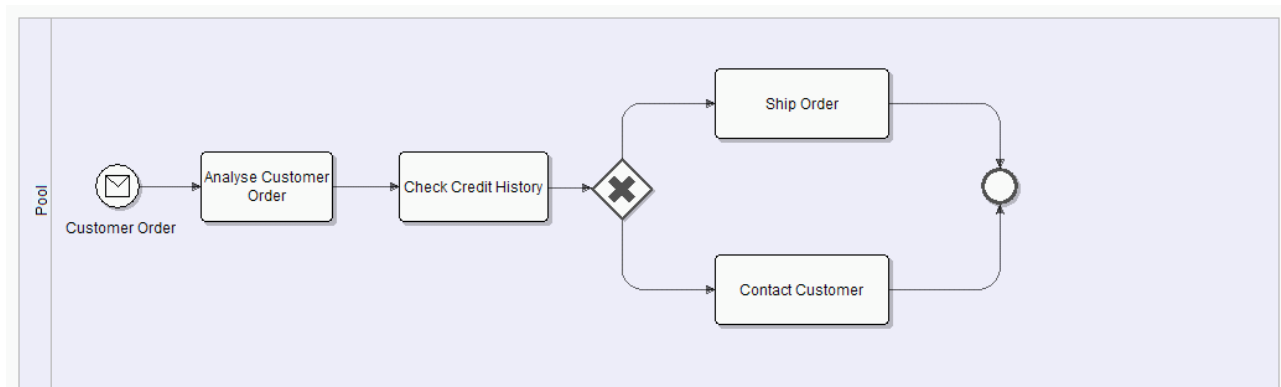


Figure 14: Simple Business Process [Own production]

These candidate services are conceptual specifications of functional requirements. The candidate services may however be impacted by opportunities for re-use. By performing a bottom-up matching of existing asset against the functional requirements, opportunities for re-use may be found. Deciding to re-use existing asset may imply re-factoring of existing services and/or changing the requirements of service candidates.

In step 4 the requirements are specified. The requirements are an amalgamation of the documentation obtained in the previous steps. It may however be necessary to define additional requirements, such as additional constraints or additional non-functional requirements. These non-functional requirements may be driven by both business and technology issues. The word *additional* should be taken very literal. The enterprise should define which types of non-functional requirements that should be captured during the process. Moreover, non-functional requirements should be captured as early in the process as possible. Otherwise, there is a risk, that the non-functional requirements captured in step 4 just becomes a “bucket” of random requirements.

Finally, in step 5 the service candidates will be specified. Specification does not concern the implementation details of the service, but rather the service interface and the service data model. The service requirements defined in step 4, along with the service interface and the service data model forms the realisation contract.

There is however an important caveat to using this procedure. While using candidate services as inputs to service specification is valuable, the implicit granularity defined by the candidate services should not be taken as gospel truth. What is the right service granularity? In theory, the granularity of services and capabilities could be derived directly from the domain decomposition, i.e. by making a direct one-to-one relationship between activities and business services.⁶⁰ But would this necessarily lead to an optimal service design? Steve Jones tells “*Why BPM screws up SOA*”:

“This is one of the big challenges of BPM and SOA in that if you start with BPM, which is about co-ordinating steps, then suddenly every service looks like a step. I've seen this problem on several occasions now, and heard it repeated by many others so it looks to be pretty endemic in BPM driven solutions.

[.....]

the SOA-RM says 'A service is a mechanism to enable access to one or more capabilities', so it is possible for it to be a single capability, but that is certainly not the only, or indeed the most likely, number of capabilities in a service.”

[Jones]

Steve Jones certainly has an important point; it is not advisable to derive service granularity directly from activities. Non-functional requirements always acts as constraints on the service design. An obvious non-functional requirement is re-use. It can often be valuable to define a broader functional scope for the service, than that required by an individual use-case. It may f. ex. be more appropriate to define an *order service*, rather than an *order entry* service in order to create cohesive services. It would also be a problem to ignore other non-functional requirements, such as performance or security considerations, which places restrictions on granularity.

⁶⁰ And by mapping the tasks encapsulated by activities to capabilities in services.

This is not to say, that BPM should not be used for service design. Rather, BPM is very valuable as it provides opportunities for aligning needs and capabilities. But service design should attempt to balance all three types of requirements, as can be seen depicted in figure 15. The three parts are linked together in a triangle on purpose, as each of the three factors have the ability to impact each other.

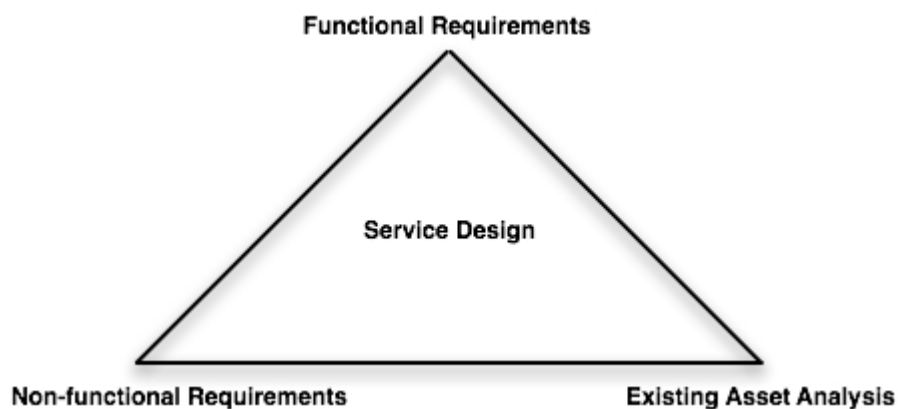


Figure 15: Service Specification Requirements [Own production]

The observation that these three requirements must be balanced against each other has some major implications. The first implication is that service identification and specification must be an iterative process, and thus less sequential than the physical layout of the figure 13 may suggest. Following a strict top-down approach would not lead to the development of the right requirements. Instead, a combined top-down and bottom-up approach is needed. This also means, that identifying and specifying services are not in themselves orthogonal activities. Business and IT will have to work closely together in order to develop the requirements, thus becoming able to specify the right services. The traditional pattern of capturing business requirements and then designing systems according to these requirements, would clearly lead to a less-than-optimal service design.

4.3.3 Service Change

A main purpose of adopting SOA is to enable looser coupled architectures. This would help enterprises move towards more dynamically re-configurable business processes. But an important problem in relation to coupling remains; even if we are able to eliminate all artificial dependency by promoting loose coupling, real dependency (or coupling) will continue to exist. An interaction between two service participants will always be done with a particular purpose in mind, as both provider and service are looking to achieve “*real word effects*”. [SOA-RM:18] It should be noted, that service-to-consumer coupling in composite service scenarios will be considered a variation of consumer-to-service coupling, as they exhibit similar problems.

Consumer-to-service coupling can be either be in the form of syntactic dependency (such as addressing, binding, or contracts) or in the form of semantic dependency (the implied meaning of the data or the meaning of the interaction between consumer and provider). [Lothka] For the most part, syntax coupling is handled pretty well in SOA. Standards that pertains to SOA (such as XML, XSD, WSDL), registries (UDDI), and communication brokers (ESB) all contributes to the reduction of syntactical coupling.

Things are however far worse in SOA in regards to managing semantic coupling. The SOA Reference Model states that, “*The primary task of any communication infrastructure is to facilitate the exchange of information and the exchange of intent*” [SOA-RM:17] This statement indicates, that there are both information elements, as well as a behavioural elements involved in service invocations. The SOA Reference Model only briefly touches upon the potential issues on semantics in SOA, but does nothing to address them. Realising, that service semantics was mostly absent in the original SOA Reference Model, work has been done to create a new *Reference Ontology for Semantic Service Oriented Architectures* (SSOA). The Semantic SOA Reference Model is however still work in progress at the time of writing this. The current incarnation of the document has reached the status of *Release Candidate 11*. The problems of behavioural and information coupling will be detailed in the next parts of the chapter.

4.3.3.1 Behavioural Coupling

Because service consumers are designed to interact with a service, there is dependency on the service; there is an actual meaning to the service interaction, whether this meaning is explicitly stated or not. With simple services, the behavioural dependency is trivial. Take a *currency conversion service* as an example. There is a small set of behaviour attached to such a service, and the exact meaning of the service is almost self-evident. It would also seem reasonable to expect no negative side-effects to be produced by invoking the simple service. The problem is orders of magnitudes larger, when looking at the coarse-grained business services; these services are expected to perform a much wider range of actions. A composite *order entry service* would f. ex. perform actions such as matching the customer against the customer database, perform look-up of the products on the order, calculate prices etc. before adding the order to the back-end system. Clearly, the client would have to make more assumptions about the behaviour of such a service, compared to that of the simple services. Thus, the problem of behavioural coupling correlates with the functional scope of the service.

The service contract is one of the most important tools for achieving loose coupling in SOA. A service contract, will typically comprise a number of different *service description documents* [Erl, 2008:126]. The *technical service contract* is a subset of the main service contract, comprising descriptions that defines the technical interface of the service, using now familiar standards such as WSDL and XML Schema. [Erl, 2008:126] But WSDL and XSD mainly provides logical representation, and does not explicitly deal with semantics. In an ideal world, the consumer would only have to depend on the service contract for information about the service. But there is also a semantic contract, which is often not specified. Moreover, one can not expect to have all assumptions about the service behaviour made explicit. As Dave Snowden puts it so eloquently: “*We can always know more than we can tell, and we will always tell more than we can write down*”. [Snowden:11] Figure 16 shows the difference between the explicitly stated technical contract and the semantic contract.

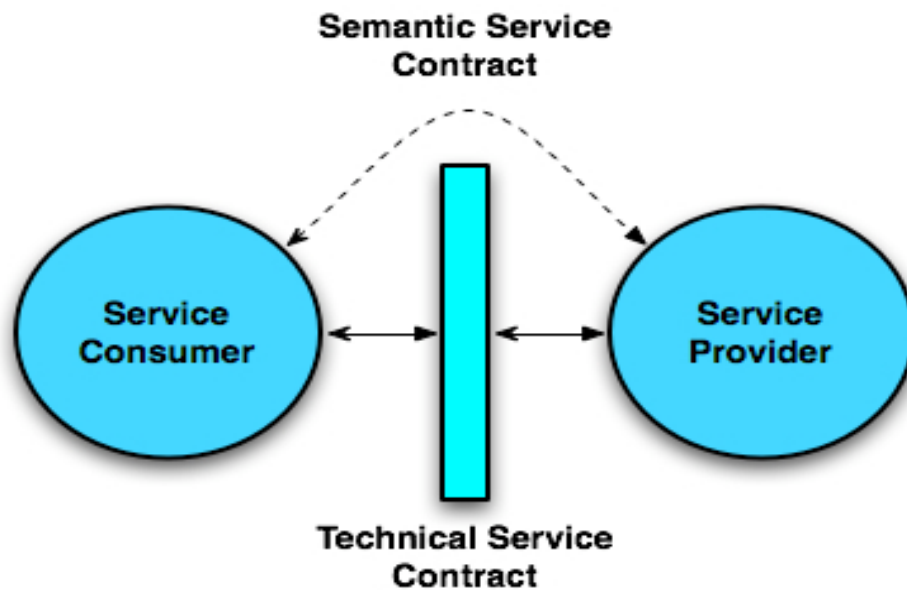


Figure 16: Semantic Service contract [Own production]

The fact that consumers cannot be completely de-coupled from the service implementation details has a number of consequences. Firstly, the semantic coupling places restrictions on the ability to change a service. Changing the implementation details of a service, would potentially break the clients assumption about the behaviour of the service, thus introducing the risk of undesirable side-effects. In shared-use scenarios, where a service is used by different consumers in different contexts, the “load” on the service increases. This is likely to place heavy restrictions on the changeability of services. This also means, that changes often have to be synchronised across several components in the architecture, as changes can have cascading effects. Changes to an entity service might f. ex. affect several other services depending on the service (or even business processes depending on the business services consuming the entity service).

Moreover, if service changes are not expressed through changes to the technical contract, then service changes might not be detected until run-time. It is clear, that unless behavioural semantics in SOA is somehow managed, then the architecture will quickly become brittle and sedimented.

4.3.3.2 Information Coupling

Another type of semantic coupling in SOA concerns the meaning of the information that the provider and consumer will exchange. In loosely coupled relationships, service participants exchange data through messages. When service participants exchange data through their interactions, they need to have a common shared understanding about the meaning of those data. Business processes relies on abstract representations of information entities, such as business objects. Therefore, an order entry service, would f. ex. need to understand the concepts of *Customer* and *Sales Order* the same way as would the business understand the concepts. While a *Customer* object may have different logical representations (or schemas) or may be stored differently in physical data-stores across the enterprise, all parties in the enterprise would ideally have a shared understanding of the *Customer* concept; they would ideally point to the same semantic definition.

Semantics of information entities are often defined inconsistently across the enterprise. A common problem is that many information entities have been established in different contexts, and for different purposes. The problem can originate horizontally, f. ex. when different legacy systems or different parts of the organisation uses different terminologies. The problem can also exist vertically, i.e. when business and IT uses different terms to denote the same thing. [Missikoff:3] There are two different types of semantic inconsistencies:⁶¹ [Missikoff:3] [Jenz:11]

- **Synonyms:** Different symbols referring to the same thing. *Postcode* and *ZIP-code* f. ex. both refers to the same part of the postal address.
- **Homonyms:** The same symbol refers to different things. A term like *person* is likely to be used differently depending on who is using the term; the legal department might use the term differently than the payroll department. Another possibility is two symbols referring two the same physical thing, but interpreting them in different ways.

⁶¹ Synonyms and homonyms can also be used to manage semantic inconsistencies

Sometimes an enterprise will partake in SOA and willingly or unwillingly ignore semantic coupling. This would lead to each service becoming a “small language” in itself, because each service would implicitly carry its own semantic definition of important concepts. This might be manageable for a small number of services, but with hundreds or even thousands of services, it would eventually turn the SOA into a “tower of Babel”. [Rugg] Generally, semantic inconsistencies can be managed through a common information model or through mapping of entities to each other.

4.3.3.3 The Impact of Coupling on Agility

Analysing the change-time phase in the service life-cycle from a perspective of coupling, has revealed some potential threats to achieving agility. It is not possible to arbitrarily change single components in the architecture without knowing the impact of these changes on other components. Disciplines like change management and impact analysis are not likely to go away just because of SOA. In fact, there is a paradox; adopting BPM and SOA results in the decomposition of monolithic structures into smaller structures. These smaller structures each need to have their life-cycle managed. While it makes for more flexible architectures, it does also in some way increase complexity. This conclusion gives rise to two observations.

Firstly, the enterprise needs to be able understand the dependencies between the components in the architecture. Architecture is about the components, their relationships and their relationship to the environment. It will be impossible to gauge the impact of modifying a component in the architecture, unless the relationship to other components in the architecture can be understood. This realisation makes it evident, that managing meta-data becomes a key discipline for the agile enterprise. The second observation is closely related to the first. Because, components in the architecture can not always be changed unitarily, then it follows, that changes may need to be synchronised across multiple components. So not only should it be possible to understand the relations between components, but it should also be possible to support some kind of versioning of multiple components. This would allow changes to multiples components to be developed and deployed in unison fashion, i.e. “releases” of changes can be defined. These two problems will be addressed in the next chapter, where enterprise architecture will be introduced.

4.4 Is BPM the Business Case for SOA?

Having determined that BPM and SOA are not orthogonal entities, it is time to turn the attention towards the next question: are BPM and SOA complementary or opposite entities? I will pose the question: “Is BPM the business case for SOA?”. Business cases are often the means, through which enterprises makes decisions about investments. There is an ongoing discussion about the '*business case for SOA*'. Ismael Ghalimi bluntly states that: “*Service Oriented Architecture (SOA) is a solution in search of a problem*”.⁶² [Ghalimi, 2006] He certainly raises important questions about the value of SOA.

The main benefits of SOA can be divided into intrinsic and extrinsic benefits. Intrinsic benefits should be understood as those benefits directly obtainable by adopting the SOA paradigm. Intrinsic benefits can be said to be about improving the efficiency of the IT-function, such as the ability to reduce integration and development costs, as well as reducing the time to market of integration and development. The main factors affecting intrinsic benefits is the promotion of asset re-use, improvements in productivity by use of models and improved flexibility by establishing more loosely coupled IT-architectures. Intrinsic benefits are typically reaped in tactical SOA projects. A likely tactical SOA scenario could be the IT-function using SOA in order to solve development and integration challenges, local to the IT-function.

Extrinsic benefits are about the effectiveness of the enterprise as a whole, and not just the IT-function. Here, SOA is regarded as an enabler of change in the enterprise. SOA promises not only to enable a more loosely coupled IT-architecture, but also that the looser coupling between business architecture and IT-architecture makes it possible to perform business process changes more dynamically. Extrinsic benefits are typically more strategic in nature, because SOA is used to make the enterprise, and not just the IT-function, responding faster to changes in the environment. The main benefit of BPM-SOA should thus be seen as the possibility to perform decomposition of functional silos, i.e. separating the business logic from implementation details.

⁶² There is a caveat to Ismael's remark; SOA is a paradigm – not a product, a technology or a concrete architecture. As a 'thought-pattern' or a set of guiding principles we can only use SOA to make physical implementation of these principles.

The effects of intrinsic vs. extrinsic benefits can be seen in table 7 below.

Intrinsic benefits	Extrinsic benefits
Reducing cost of application integration	Reducing cost of Business process change
Reducing time of application integration	Reducing time of Business process change
Reducing cost of application development	Reducing operating costs as a result of
Reducing time of application development	automation

Table 7: Benefits of SOA [Own production]

Is BPM the business case for SOA? The answer must be yes; process improvement projects provides exactly the kind of leverage that SOA needs. Moreover, the positive effect of SOA in combination with BPM will be of increasing strength over time, as the enterprise will possess a larger and larger portfolio of re-usable services, which can be used to re-configure existing business processes or to deploy new ones. Strategic SOA can only be achieved by being connected to business changes. Tactical SOA is for efficiency, strategic SOA is for effectiveness.

Furthermore, using BPM for domain decomposition allows business services to be well aligned with business goals and policies. This is a form of functional integration. But establishing an infrastructure for BPM and SOA requires a long-term vision that reaches far beyond that of the individual process improvement project. To provide the vision for doing so, the enterprise should see BPM-SOA in a strategic light, rather than just a tactical one. The purpose of adopting BPM-SOA should be seen as a way to achieve strategic fit, i.e. to tie the internal capabilities to the external positioning of the enterprise.

4.5 Chapter Summary

The purpose of this chapter was to analyse the relationship between BPM and SOA. I started out by describing the vision of BPM-SOA, as it was often popularised by vendors. The nirvana of BPM-SOA would mean that a business analyst was able to manage the entire business process life-cycle without intervention from IT.

I then proceeded to view BPM-SOA from two different perspectives. The first perspective was the model-driven perspective, in which BPM-SOA was seen as a paradigm that emphasised the use of models, rather than code. Following a proposed modelling value-chain for BPM-SOA, this value-chain was analysed for semantic gaps. It was found that there were still a semantic gap between process modelling and process execution. With some limitations however, this gap was however possible to bridge. Future modelling languages may provide better opportunities for closing the gap, but the gap represents a fundamental cleft between business and IT.

It was found, that whenever appropriate discoverable and re-usable services was available to the business analyst, then it would be possible to dynamically re-configure business processes. But it was also found, that it is not possible to automatically bridge the gap between business modelling and service modelling. As such, managing the business process life-cycle will continue to require the collaboration of business analysts, solution architects and system developers. Because of this, BPM and SOA cannot be considered orthogonal entities.

The methodology-driven perspective were focused on bridging semantic gaps. BPM was proposed as a tool that could be used for domain decomposition. Business process analysis and design would both provide important insights as to process related meta-data and the work to be done during the process. This lead to a conceptual model for identifying and specifying services. The process seeks to balance functional requirements, non-functional requirements and re-use of existing assets. The service design phase should thus be considered iterative. Moreover, business and IT will have to work closely together to determine the right requirements. The capture-then-design pattern must be abandoned.

The change-time phase of the service life-cycle was analysed from a perspective of coupling. It was found that semantic coupling, in the form of behavioural and information coupling, would absolutely have to be managed. This led to the observations that meta-data management would be a critical competence, and that the ability to develop and deploy components in the architecture in a synchronised way would be vital too.

Finally, it was asked if BPM is the business case for SOA. It was found that BPM certainly provides an important leverage for SOA. In this sense, BPM and SOA are complementary. But it was also found that the vision to establish BPM-SOA should be based in strategic considerations. As such, the decision to adopt BPM-SOA should always be seen as an attempt at creating strategic fit.

The final conclusion is that the working hypothesis of BPM-SOA being orthogonal but complementary must be rejected. BPM and SOA should instead be considered complementary but overlapping entities.

Chapter 5 – Contribution of Enterprise architecture

In most of the previous chapter, the relationship between BPM and SOA was seen in a very project-oriented light. The focus was on using business services as an abstraction layer between business and technology. However, building an architecture for BPM-SOA requires investment, planning and management that goes far beyond the scale of individual BPM-SOA projects. The purpose of this chapter is to analyse how EA can help meet these demands. TOGAF will be used as the base for the analysis. The primary reason for this is that TOGAF to some degree can be considered a candidate for a de-facto EA methodology.⁶³ In cases where references to a taxonomic framework will be needed, I will reference the Zachman Framework. The two frameworks are illustrated in appendix D & E.

The chapter will fall in five main parts. I will start out by briefly introducing BPM and SOA in the context of the TOGAF Architecture Development Method (ADM). This is purely for setting the stage for the remainder of the chapter.

Then I will proceed to analyse how EA affects agility; in some ways EA will contribute to agility, but in other ways EA will also be an impediment to agility.

In the third part of the chapter, I will contrast the approach prescribed by the TOGAF ADM to that of the three key disciplines for realising a “*Foundation for Execution*” (as described by Ross, Weill & Robertson in their “*Enterprise Architecture as Strategy*”). Their work provides some unique perspectives, which can help nuance some of the problematic assumptions behind the TOGAF ADM approach.

In the fourth part of the chapter, I will integrate what has been learned from the ADM, with what has been learned from the Foundation for Execution approach. The purpose is to modify the ADM, thus establishing an integrated approach to architecture development, which can be used for managing BPM-SOA for agility.

⁶³ Given the commonalities among EA frameworks, many of the points in this analysis, should hold across different methodology oriented frameworks.

In the final part of the chapter, I will argue that a common language for EA is needed in order to realise the integrated approach. The role of this language is to connect the many different EA artefacts. I will also look at some of the challenges of creating such a language. Finally, I will sketch out a meta-model, and some high-level requirements for the language. This will set the stage for the next chapter, where I will further investigate the opportunities for creating such a language.

5.1 Architecture Development in the Context of BPM-SOA

The first part of this chapter will detail how BPM and SOA fits into the ADM.

Even though the TOGAF ADM is generic in nature, and does not specify any architectural style, there are some challenges to using TOGAF for SOA.⁶⁴ Certain enhancements are to be made to the ADM by adding or modifying objectives, inputs, steps, and outputs of the individual phases. Therefore, TOGAF has launched the SOA/TOGAF Practical Guide Project. [TOGAF, 2006] The aim of the project was to deliver guiding principles concerning the use of the ADM for SOA, which would be considered “*good enough*” and “*practical*” to the architecture practitioner. [TOGAF, 2006:1] The work is primarily aimed at making adjustments to the *Preliminary Phase* and Phase A-D in the ADM. The “*Delivering SOA with TOGAF*” [Dico] presentation is an output of the Practical Guide Project, and details the changes needed to be done against the ADM. Work from this project will be used (and specifically referred to) during the next parts of this chapter.

⁶⁴ Better support for SOA is supposedly one of the design goals for TOGAF 9, which is in the making. So far not many details about TOGAF 9 has been released publicly. There has not been communicated any time of release either

One key question concerning the use of the ADM for SOA is to understand, where SOA services fits in the ADM cycle. TOGAF states about the Application Architecture that: *“applications are not described as computer systems, but as logical groups of capabilities that manage the data or business Architecture”*. [TOGAF:67] This definition sounds remarkably similar to the definition of services in SOA. Business services in particular, consists of collections of capabilities used to support business processes, and entity services are collections of capabilities used to manage data. Applications in contrast, are implemented by composing or orchestrating services. [Dico:10] In this sense, the phase C of the ADM can both be referred to as Service Architecture and Application Architecture. [Dico:10-11]

As for integrating BPM into the ADM cycle, there are no official recommendations. The business architecture phase do however contain many of the same elements, that would be found in a structured approach to BPM. Recalling the description of the 7FE BPM framework given earlier in this thesis, it is fairly obvious that the process architecture overlaps very much with the ADM business architecture. In particular, there is a great overlap of modelling artefacts between BPM and business architecture, such as: [TOGAF:50-51]

- Business goals and objectives
- Business functions
- Business services⁶⁵
- Business processes
- Business roles
- Business data model

Furthermore, both the ADM and BPM supports the notion of modelling *as-is* and *to-be*.

⁶⁵ As recalled from the discussion in chapter 3, the notion of Business Services in TOGAF is different from the notion of Business Services in SOA.

5.2 Enterprise Architecture and Agility

The purpose of this section is to investigate how the TOGAF ADM impacts agility. It will be argued, that the ADM can both be seen as an enabler of agility, as well as an impediment to agility.

5.2.1 TOGAF and Alignment

As Scott Bernard notes, EA is unique in providing enterprise-wide thinking about resource utilisation. [Bernard:61] It is especially qua this unique role, that EA has the potential for impacting agility in a positive way. In the following it will be the argued, that EA primarily promotes alignment, but that this alignment is needed for enabling sustained agility.

The ADM provides both strategic fit and functional integration. Figure 15 depicts the ADM phases A-D from the perspective of alignment, rather than through the regular cycle view. The figure is organised according to the the same dimensions used in the Strategic Alignment Model by Venkatraman & Henderson.⁶⁶ Phases A-D places emphasis on the creation of Baseline and Target Architectures, which are important in regards to planning for alignment. But whereas phase A-D are concerned with creating conceptual solutions to requirements, phase E-H (Opportunities and Solutions, Migration Planning, Implementation Governance, and Architecture Change Management) concerns the actual implementation and management of the architecture.

⁶⁶ See figure 1 on page 20.

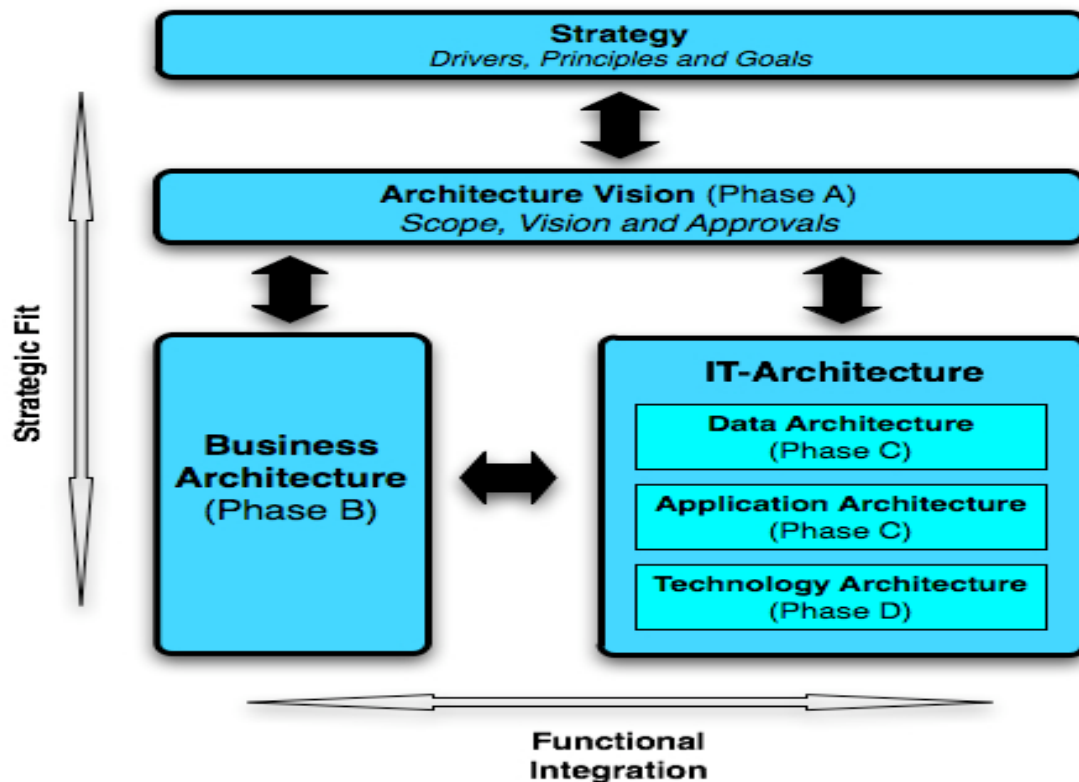


Figure 17: Alignment Model for the ADM [Own production]

The ADM provides strategic fit by having the architectural work being aligned with the strategy. The foundation is laid in the *Architecture Vision* phase of the ADM (Phase A), which takes strategy as input, including goals, drivers, and principles. The initial Baseline and Target Architectures for the business, application, data and technology domains are created as output from the *Architecture Vision*. The initial Baseline and Target Architectures are then further elaborated upon during phase B-D. In this way, the enterprise strategy sets the context for developing the architecture. Requirements can be developed through what TOGAF calls *Business Scenarios*, which are used to identify and understand business needs. [TOGAF:375] Thus, the *Architecture Vision* sets the scope for the further work being performed in the ADM, and ties together the external positioning of the enterprise, with the internal architectural work being done in the rest of the ADM cycle.

Such an approach to creating strategic fit can provide the commitment to establish BPM-SOA. This is especially evident in the enhancements to the ADM *Preliminary Phase* and the *Architecture Vision* (Phase A) as specified in the SOA Practical Guide Project.

The following objectives have been added or changed to the Preliminary Phase: [Dico:5]

- To define the architecture principles (**including SOA principles**)
- To assess SOA readiness, maturity and define SOA adoption plan
- To setup and monitor a process (**including SOA governance framework**)

Moreover, the following objective have been added to the Architecture Vision phase: [Dico:6]

- To develop business case for SOA for this architectural style

By connecting the enterprise strategy to the architecture development process, and by taking the long-term enterprise-wide approach to resource development, it becomes possible to make more coherent decisions about the architecture. Thus, the ADM can help provide the long-term vision and the commitment to embark on the path of BPM-SOA. Moreover, the ADM serves as a structured planning tool, that ensures that the architecture work is aligned with enterprise strategy.

Functional alignment does not in itself produce agility. One can hypothesise, that there is a relationship between higher degrees of functional integration, and a reduction in complexity of the architecture. Much of the current complexity in the architecture comes from short-sighted decisions, that makes for quick adaptations, but eventually contributes to the sedimentation of the architecture. Functional integration would f. ex. typically result in a reduction of duplicated logic across the enterprise, which could make the architecture more transparent. So the improved planning that leads to functional integration, would probably also lead to the creation of architectures that are better thought out, and thus less complex to manage.

More importantly however, the ADM reflects one of the key findings from the BPM-SOA chapter pretty well; because there are still dependencies between the architectural domains, neither of them can be developed independently of the other. Thus, the ADM provides an overarching framework for developing the four architectural domains in a concerted fashion. The ADM is however very generic in nature, and more fine-grained methodologies will be called for. One obvious example would be SOAD, including the process for identifying and specifying services that was suggested in the previous chapter.

The overall impression from this investigation is that the ADM is very well suited for achieving alignment. Strategic fit in particular is important, as it provides the vision and the commitment for building the BPM-SOA infrastructure. And while functional integration does not provide agility in itself, it does provide an overarching framework for building the four architectural domains in a concerted fashion, as well as for creating less complex architectures. Such architectures should be easier to change and more sustainable.

5.2.2 TOGAF and Agility

Having seen how the ADM can certainly be an enabler of agility, especially by providing the long term vision to establish BPM-SOA, it is now time to turn the attention towards ADM as an impediment to agility. In this regard, it will be especially important to distinguish between the deliverables of the ADM methodology (architectures and EA artefacts), and the ADM as a methodology. It will be shown that certain assumptions around the ADM as a methodology, impacts agility negatively.

From a macro perspective, the ADM does appear to be a mostly top-down approach to resource development. As described earlier, the ADM starts with the big picture by establishing initial Baseline and Target Architectures, which are then elaborated upon in later phases. The top-down view is however moderated a bit; there are iterations both among the individual phases, and among the steps within each individual phase. [TOGAF:19] Figure illustrates this point.

It is also possible to re-order the phases of the ADM. [TOGAF:22] An example where the re-ordering of phases would be an implementation of a standard ERP system. In such cases, business processes are often adapted to fit the “best practices” processes defined by the ERP vendor.

An evaluation of TOGAF as a tool for agility cannot be done without considering the ADM cycle length. But it is notoriously difficult to make any exact predictions as to the cycle time of an iteration; there are too many variables at play. TOGAF does however indicate, that the first iteration is the longest, because there will be a need to generate many new artefacts for the Enterprise Continuum.⁶⁹ Subsequent iterations should be faster, as it will be possible to leverage existing artefacts. Previous iterations should also be able to cope with some of the most urgent and pressing architectural needs, which makes later iterations faster. But it is also quite obvious, that we should consider ADM cycles fairly long and at least substantially longer than the iterations in agile software development projects. [Temnenco] The ADM is however a very adaptable framework, and there several ways to shorten the cycle time if needed. For each iteration a number of decisions that affects the cycle length must be made: [TOGAF:24-29]

- **Scope:** In very large enterprises, such as those found in federated environments, performing a full ADM can almost be a mission impossible. For this reason, the architecture can be reduced to certain business sectors, functions, geographical areas, or organisations.
- **Architecture Domains:** A complete ADM contains all four architecture domains, but it is not always realistic to include all four domains in the same iteration. Thus, one or more architectural domains may be omitted (or reduced) within a particular cycle of the ADM. Business architecture should however always be present.
- **Vertical Scope:** Care must be taken to decide the appropriate level of detail in the architecture effort. Especially, the demarcation between the architecture effort and related activities like system design, and system engineering should be decided.
- **Time:** To meet time demands, a transformation can be divided across several iterations of the ADM cycle. The target architecture is then defined for the overall system, with intermediate *Transitional Architectures* in-between.

⁶⁹ The ADM however also states, that an enterprise, does not have to create a detailed architecture description in the first attempt [TOGAF:28]

The ability to shorten the ADM cycle should however not be taken as a *carte blanche* to compromise on basic EA principles. TOGAF does in particular warn against selecting a too narrow scope. The TOGAF f. ex. re-iterates a warning from the “*Practical Guide to Federal Enterprise Architecture*”: [TOGAF:26]

“It is critically important that enterprise architecture development be approached in a top-down, incremental manner, consistent with the hierarchical architecture views that are the building blocks of proven enterprise architecture frameworks. ... In doing so, it is equally important that the scope of the higher-level business views of the enterprise architecture span the entire enterprise or agency. By developing this enterprise-wide understanding of business processes and rules, and information needs, flows, and locations, the agency will be positioned to make good decisions about whether the enterprise, and thus the enterprise architecture, can be appropriately compartmentalized. Without doing so, scoping decisions about the enterprise architecture run the risk of promoting “stove-piped” operations and systems environments, and ultimately sub-optimizing enterprise performance and accountability.”

From this warning we can discern, that TOGAF certainly advocates a top-down approach, and that at least at the business architecture should be defined enterprise-wide. So even if the ADM is iterative, it would not be correct to consider it an agile methodology. This is clearly problematic; if business change cycles can be expected to be shorter than the ADM cycles, then the EA effort would then become a bottleneck. Thus, from this point of view, the ADM seems to prioritise alignment over agility.

5.2.3 Architecture Change Governance

Because the ADM is not a particular agile methodology itself, it becomes vital to determine how the ADM responds to sudden shifts in requirements and priorities. The most interesting phase in TOGAF in regards to this is the *Architecture Change Management* (Phase H). This phase involves setting up processes to monitor the environment for changes, including “*Monitor Technology Changes*” and “*Monitor Business Changes*”. A systematic process to monitor the environment for business and technology changes is an important step in achieving agility, as proactive monitoring is likely to result in faster and better detection of opportunities and threats. This is related to the concept of *early warning capability*, that was mentioned in chapter 2.

Not all changes are handled the same way. Some changes are incremental in nature and can be handled within the current ADM iteration. Other changes have wider implications and affects the entire architecture. Such changes can only be handled through a new iteration of the ADM. The ADM suggests the following structure for categorising changes: [TOGAF:113]

- **Simplification change:** A simplification change can normally be handled via change management techniques.⁷⁰
- **Incremental change:** An incremental change may be handled via change management techniques, or it may require partial re-architecting. This decision obviously depends on the nature of the change.
- **Re-architecting change:** Change that requires restarting the entire architecture development cycle.

⁷⁰ Simplification changes, could f. ex. be de-commissioning elements in the architecture, such a systems, components, or services.

The guidelines put forward by TOGAF for determining whether to manage a change incrementally or by restarting the ADM circle are fairly strict:

- **Restart:** If the change impacts two stake-holders or more, then it is likely to require an architecture re-design and re-entry to the ADM.
- **Incremental:** If the change impacts only one stake-holder, then it is more likely to be a candidate for change management.
- **Dispensation:** If the change can be allowed under a dispensation, then it is more likely to be a candidate for change management.

Under these guidelines, many BPM-SOA related projects, involving both processes- and services would require a restart of the ADM. This approach does not deal very well with unexpected changes. It is clear, that such strict Architecture Change Management policies, would be an impediment to agility.

5.2.4 Discussion: TOGAF for Agility?

As it has now been established, the ADM is a very good tool for establishing sustainable and flexible architectures. On the other hand, it has also been found, that ADM iterations were rather long and that the Architecture Change Management is inflexible. In general, it seems that the underlying assumption of the ADM is that target architectures can be based on a fairly fixed baseline; changes to the baseline are to be considered exceptions.

Figure 19 illustrates a more realistic scenario. Both Baseline Architecture and Target Architecture have been documented. In-between the Baseline Architecture and the Target Architecture iterations of the Baseline Architecture will emerge, as new opportunities and threats are being addressed continuously. Many of these solutions will impact several stake-holders. The Target Architecture always operates on a moving Baseline Architecture.

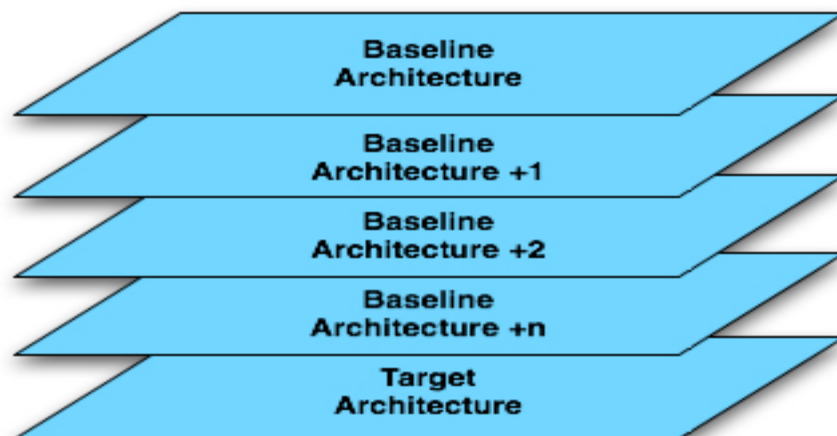


Figure 19: Architecture Change [Own production]

In agile software development practices, the inherent unpredictability of requirements are being managed through the use of iterations. So why are iterations less efficient in the ADM? The answer is that because the iterations in TOGAF are much longer, and because the enterprise is always moving against a baseline that will itself evolve. A rigid interpretation of the ADM would just compartmentalise the architecture development into a series of sequential steps. If the TOGAF ADM approach was to be taken literally, it would stifle the enterprise. The enterprise could create highly efficient architectures, but at the cost of agility.

In reality, there are those changes that an enterprise will know ahead of time, and those changes that an enterprise knows will happen eventually. But the most difficult changes, to deal with, are those changes that cannot be foreseen. [Bloomberg & Schmelzer:5] The ADM is very well suited for planning change, and planning for change. Less so for managing the unforeseen. It is obvious, that a more balanced approach to enterprise architecture development is needed.

5.3 Building a Foundation for Execution

In this part of the chapter, the approach taken by the ADM, will be contrasted against the approach suggested by Ross, Will & Robertson in their “*Enterprise Architecture as Strategy*”.⁷¹ They describe a Foundation for Execution, which is the “*IT infrastructure and digitized business processes automating a company's core capabilities*”. [Weill, Ross & Robertson:4] This idea would fit well with the establishment of a service oriented infrastructure and a process architecture, leading to the digitisation of business processes.

The authors argues, that some companies perform better than others, because they have a Foundation for Execution. These companies has made IT an asset, rather than a liability. Through a survey of 103 U.S. and European companies, they discovered, that 34% of these companies had digitised their core processes.⁷² The companies excelled by having higher profitability and got more value form their IT investments. Yet, the companies that had digitised their core processes also had IT costs 25% lower than those who did not. [Ross, Weill & Robertson:2] Moreover, they argue that these companies are also more agile.⁷³ These companies, they argue, have successfully built a Foundation for Execution.

Their approach does in some way parallel the ADM approach, but does also provide some important nuances. The knowledge obtained in this part of the chapter, will be used in the next part of the chapter, establishing a model illustrating the integrated approach.

71 Which was briefly introduced in the final part of the last chapter.

72 There is no mentioning to which degree, the companies in the survey among 103 U.S. and European companies, had redesigned business process, along with the digitisation of their processes. This would however seem plausible.

73 Their explanation of why these companies becomes more agile are at times a bit awkward. They f. ex. speculate that: “...*having a digitized Foundation for Execution probably enabled managers in these companies to spend more time focusing on what products would succeed and then bringing those products to market*”. [Ross, Weill & Robertson:2] In similar vein it is said: “*Managers cannot predict what will change, but they can predict somethings that won't change. And if they digitize what is not changing, then they can focus on what is changing*”. [Ross, Weill & Robertson:12]

There are three suggested key disciplines, for effectively building a Foundation for Execution: [Weill, Ross & Robertson:8-9]

- **Operating Model:** The Operating Model contains a commitment as to how the company operates. Defines the necessary level of business process integration, and standardisation for delivering goods and services to customers. The operating model defines which core processes and systems to standardise, and which to integrate. Integration of processes allows for end-to-end processing, and a single interface towards the customer, but forces a common understanding of data across the company.
- **Enterprise Architecture:** The enterprise architecture is the organising logic for business processes, and IT infrastructure. As such, the operating model is implemented via enterprise architecture. EA provides the long-term view on the enterprise, including processes, systems, and technologies. It is exactly the long-term view that allows companies to build capabilities, rather than just focus on immediate needs.
- **Engagement Model:** The system of governance mechanisms to ensure that business, and IT projects achieve both local and companywide objectives. The Engagement Models provides linkages between senior-level IT decisions, such as project prioritisation, and company-wide process design, and project-level implementation decisions.

The three key disciplines will be detailed below, and contrasted against the ADM approach.

5.3.1 The Operating Model

The first step in creating a Foundation for Execution is to formulate an Operating Model. The Operating Model is essentially a commitment on how to perform the business. [Weill, Ross & Robertson:26] The Operating Model consists of two different dimensions; the level of business process integration across business units, and business process standardisation across business units. This segmentation leads into a classic two-by-two matrix structure, which can be seen in figure 20.

Business process integration	Coordination: <ul style="list-style-type: none"> • Shared customers, products or suppliers • Impact on other business unit transactions • Operationally unique business units or functions • Shared customer/supplier/product data • Consensus processes for designing IT infrastructure services • IT application decisions made in business units 	Unification <ul style="list-style-type: none"> • Customers and suppliers may be local or global • Globally integrated business processes often with support of enterprise systems • Business units with similar or overlapping operations • Centralized management often applying functional/process/business unit matrices • High-level process owners design standardized business processes • Centrally mandated databases • IT decisions made decentrally
	Diversification: <ul style="list-style-type: none"> • Few, if any, shared customers or suppliers • Independent transactions • Operationally unique business units or functions • Autonomous business management • Business unit control over business process design • Few data standards across business units • Most IT decisions made within business units 	Replication: <ul style="list-style-type: none"> • Few, if any, shared customers • Independent transactions aggregated at high level • Operationally similar business units • Autonomous business unit leaders with limited discretion over processes • Centralized (or federal) control over business process design • Standardized data definitions but data locally owned with some aggregation at corporate • Centrally mandated IT services

Business process standardization

Figure 20: Operating Models [Adapted from Weill, Ross & Robertson:29]

The basic premise for using the Operating Model as guidance for the architecture effort is that an Operating Model is less volatile than strategy, and thus provides a firmer basis for creating the Foundation for Execution. The authors argues, that “[because of changing strategic directions]...*strategy rarely offers clear direction for development of stable IT infrastructure and business process capabilities*” [Weill, Ross & Robertson:25] Formulating the Operating Model thus concerns formulating a very high-level set of requirements, which drives future business and IT initiatives. Since the Operating Model predates strategy, selecting the Operating Model also predates the enterprise architecture effort, and is thus out of scope of the ADM.

There is a subtle relationship between the Operating Model and enterprise strategy. At first glance, it seems like the Operating Model is completely distinct from strategy. By basing the foundation of execution on the Operating Model, rather than the strategy, it is easier to achieve strategic agility. The argument is, that if an architecture is based on achieving strategic fit, then the strategy and the architecture is likely to be tighter coupled, than had the Operating Model been used to formulate the architecture requirements. On the other hand, the authors also point out that the Operating Model will make limits as to which strategies that can be pursued. This notion is expressed in this way: *“Thus, the Operating Model is a choice about what strategies are going to be supported”* [Weill, Ross & Robertson:26] But this limitation also goes the other way around. In most cases, enterprises already have existing strategies and existing architectures, which will limit the opportunities for selecting an Operating Model. Depending on circumstance, there is a risk that the Operating Model becomes more of a description, than a prescription. The Operating Model and strategy are far more intertwined than what may seem at first glance.

Despite these observations, the Operating Model do indeed pose some important questions on how to scope a BPM-SOA effort; the chosen level of standardisation and integration are decisive factors when choosing whether a BPM-SOA effort should span several business units or not. Figure 21 depicts the same matrix structure as seen in figure 20, but now describes how the different Operating Models affects BPM, SOA and the data model.⁷⁴ The model is in part based on [Malik].

⁷⁴ The table is created based on the simple premises, that integration (all things equal) requires a shared information model (coordinated or common), whereas standardisation requires centralised process management, and central SOA

Business process integration	Coordination: <ul style="list-style-type: none"> • Coordinated data model • Multiple service-oriented architectures • Multiple process architectures • Shared business services 	Unification <ul style="list-style-type: none"> • Shared data model • Shared service-oriented architecture • Shared process architecture • Shared business services
	Diversification: <ul style="list-style-type: none"> • Decentral data model • Multiple service-oriented architectures • Multiple process architectures • Few shared business services 	Replication: <ul style="list-style-type: none"> • Decentral data model • Replicated service-oriented architectures • Replicated process architectures • Few shared business services

Business process standardization

Figure 21: The Effect of Operating Models on BPM-SOA [Own production]

The Operating Model certainly provides food for thought, as on how to scope the architecture development in the Architecture Vision phase.

5.3.2 Enterprise Architecture

But enterprises needs more than an Operating Model to guide their architecture effort. As the authors says: “[Companies] looking to build a strong Foundation for Execution need more detail than the Operating Model provides – they need an enterprise architecture to guide their efforts” [Ross, Weill & Robertson:46] EA is used to implement the Operating Model. The word *implement* should perhaps be taken a bit lightly here, as the EA that they describes only concerns the high-level logic for business processes and IT capabilities. More precisely, “the enterprise architecture delineates the key processes, systems, and data composing the core of a company’s operations”. [Ross, Weill & Robertson:46-47] One could say, that the EA here described here, roughly equates to the Architecture Vision (phase A), and the four architectural domains (phase B-D) in the ADM. The notion of business design (as seen in ADM phase B) does however seem to be de-emphasised. The main focus is automation, not process design.

The perhaps most surprising aspect of this approach to EA is that the EA effort is not at all driven by strategy, but only by the Operating Model. This is remarkably different from the approach devised by the ADM, which takes strategy, including goals, drivers, and principles as input. The EA approach described here, thus only defines the core capabilities of the enterprise. EA in this context does not aim to address any specific needs of the enterprise. Another interesting aspect is that EA is not concerned with the actual implementation. There is no methodology attached to this approach to EA. This view on EA, is in some way quite limited compared to that of the ADM.

5.3.3 IT Engagement Model

The final key discipline needed for implementing a Foundation for Execution is the IT Engagement Model, which is defined as “*the system of governance mechanisms assuring that business and IT initiatives achieve both local and company-wide objectives*”. [Ross, Weill & Robertson:118-119]

The Engagement Model is thus responsible for updating and evolving the enterprise architecture. The IT Engagement Model is used to co-ordinate companywide (enterprise-wide) business unit, and project activities, [Ross, Weill & Robertson:118-119] and includes company-wide IT-governance, project management, and linking mechanisms connecting the project-level activities to overall IT-governance. The Engagement Model includes work that would be performed during Migration Planning (phase F), Implementation Governance (phase G), and the Architecture Change Management (phase H) of the ADM.⁷⁵ One key task of the Engagement Model is to implement the requirements as formulated by the enterprise architecture. The Engagement Model is also valuable in the sense, that it provides the link between solution specific projects and the overall architecture.

The main difference compared to the ADM is that the EA work is separated from the actual Migration Planning, Implementation-, and Architecture Change Governance. This frees the governance model to pursue both enterprise-wide, and more local projects at the same time. The tight link between the architecture development cycle and the architecture governance, was exactly one of the problems with the ADM.

5.3.4 Is the Foundation of Execution the Better Model??

Having seen the approach to building a Foundation for Execution, it can be asked, if this approach is better than using the ADM for BPM-SOA? By now the characteristics of their approach stands clear; the focus on building capabilities rather than solutions, the separation of EA and the Engagement Model, and the use of an Operating Model to guide the EA effort.

⁷⁵ The Engagement Model could be considered a broader concept, than the content of phase F + G + H in the ADM, as the Engagement Model concerns the enterprise-wide IT-governance

In [Doucet et al.] the authors argue that there are three modes of EA, representing progression in thought and practice of EA. [Doucet et al.:1] The first level (and thus lowest level) is called *Foundation Architecture*. The primary purpose of this mode of EA is to align business and IT, by capturing information about the business, and then design IT according to business requirements. Figure 22 illustrates this type of EA. The approach to EA presented by Ross, Weill & Robertson can be classified as Foundation Architecture. Their approach seems primarily concerned with automating business processes. Business design is de-emphasised, and the Operating Model is used for creating the high-level requirements. The Foundation of Execution can in many ways be considered rather IT-centric.

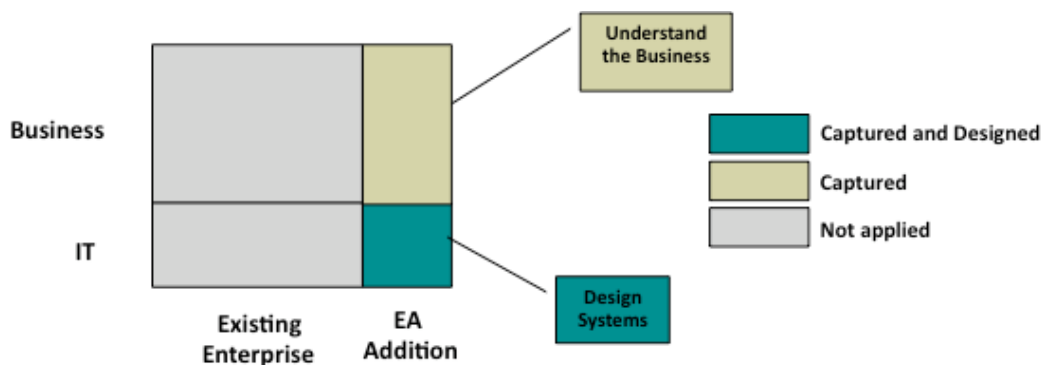


Figure 22: Foundation Architecture – Aligning Business and IT [Doucet et. al:5]

To be able to capture and design both business and IT is a part of the intent behind adopting BPM-SOA to begin with. Thus, this is closer to the second mode of EA, which is called *Extended Architecture*. (see figure 23). In this mode of EA, the focal point is to engineer enterprises from an integrated strategy, business, and technology perspective.

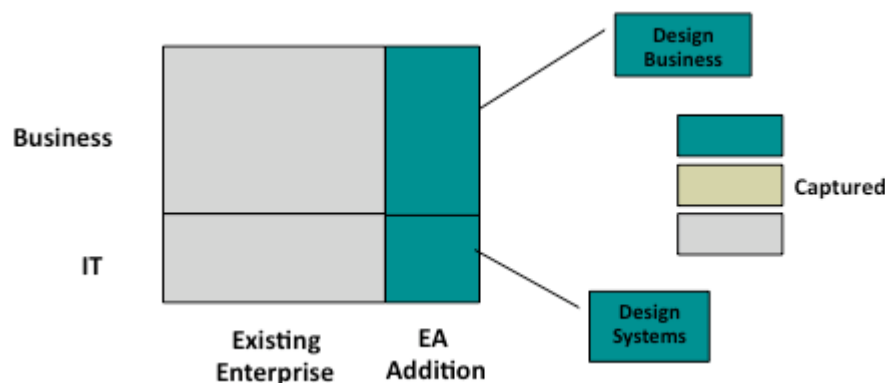


Figure 23: Extended Architecture [Doucet et al.:6]

A quote highlights the difference between the two approaches: [Doucet et al.:5]

“Whereas Foundation Architecture used architecture methods and tools to capture business requirements in order to design better IT systems, in the extended approach architecture methods and tools capture strategic goals and related business requirements in order to design the enterprise.”

It is obvious, that the ADM emphasises an integrated strategy, business and technology approach to EA to a higher degree than the Foundation for Execution does. So the ADM is in some ways closer to the ideal behind the Extended Architecture, compared to that of the Foundation for Execution. On the other hand, the business design aspects of the ADM are far from perfect. While the ADM does emphasise business design, there is still an air of “capture-then-design” over the ADM. The ADM does f. ex. not attempt to establish a bottom-up dialogue, such as required by the SOAD paradigm. The inclusion of BPM and SOA specific methodologies will therefore modify the rather top-down approach to architecture development. Even if the ADM does emphasise both strategy, business and technology, the three domains does not seem to well integrated.

When it comes to delivering practical guidance, the ADM approach does however outshine the approach by Weill, Ross & Robertson. The key disciplines for creating the Foundation for Execution completely lacks tools and methodologies for performing the architecture work. In contrast, by providing the ADM, TOGAF excels in this area. The overall conclusion is that the Foundation of Execution holds some appealing aspects, but the approach to architecture development is also found wanting in many respects. As such, it can be considered more of a thinking tool concerning EA, rather than a concrete approach to EA.

The next part of the chapter will be spent on attempting to bridge the teachings from the ADM, with the teachings from the Foundation of Execution.

5.4 The Integrated Approach

The purpose of this section of the chapter is to devise a modified Architecture Development Method.

The main idea is to preserve as many elements from the ADM as possible. Although there are certainly problems to be found in the ADM, there are also a lot of strengths to be found; the ADM is indeed a very practical approach to architecture development. So an overall design goal of the model is to preserve as many core ideas from the ADM as possible, while still being able to alleviate the problems that causes the ADM to stifle agility. The modified architecture development method is illustrated in figure 24.

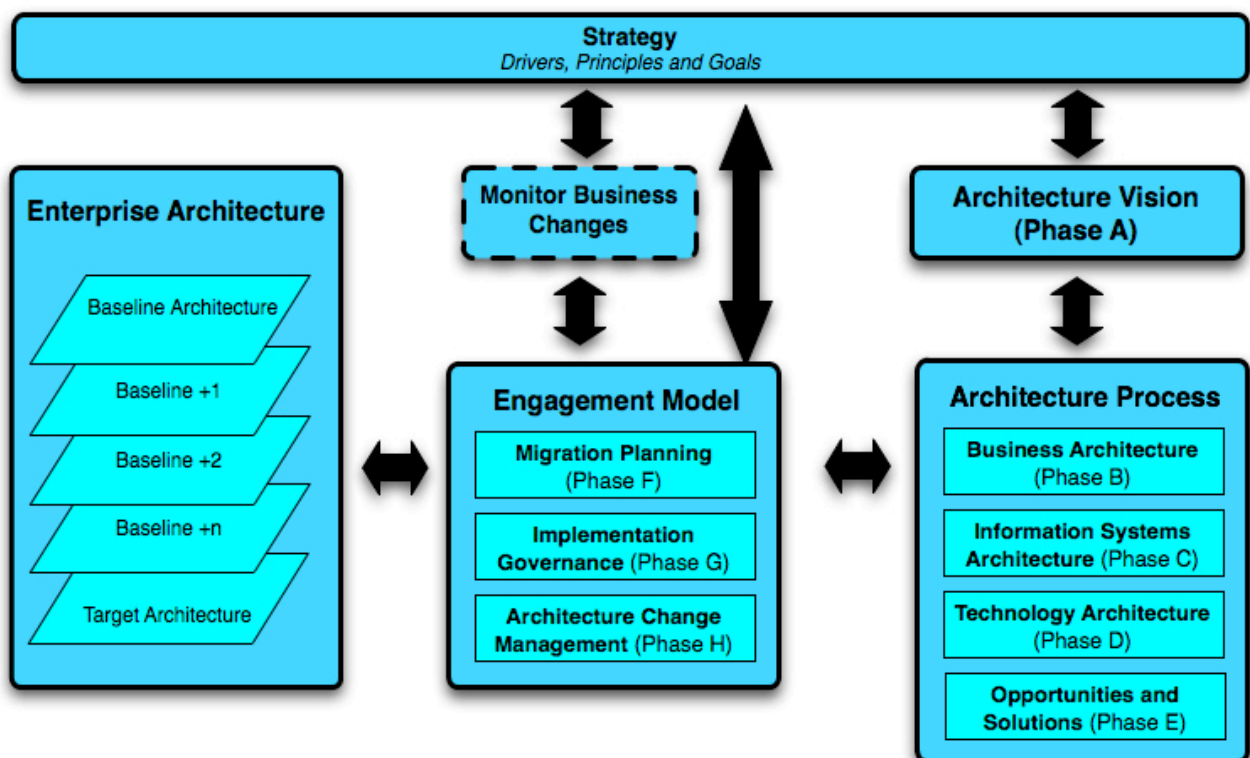


Figure 24: Modified Architecture Development Method [Own production]

There are several overarching principles behind the model design. The first principle is to establish a separate Engagement Model. The Engagement Model will define important governance processes, principles and incentives. One of the key findings from the previous analysis of the ADM was, that the architecture governance (phase E, F, G & H) was to closely coupled to the ADM cycle. The separation of the Engagement Model from the rest of the architecture work, as suggested by Ross, Weill & Robertson, was clearly a more flexible approach. By de-coupling the architecture governance from the ADM cycle, it becomes possible to pursue long-term proactive planning, as well as performing more short-term reactive adaptations to the architecture. Moreover, in this model the Engagement Model is guided by strategic concerns, which will provide the context for developing the architecture. The “*Monitor Business Changes*” provides input to the Architecture Change Management phase. Business change drivers can thus invoke the change management process, which can then affect the architecture development process.

The Engagement Model is used for updating and evolving the enterprise architecture in exactly the same way as seen by the Foundation to Execution approach. The way the enterprise architecture is being developed in this model however, will be different from that of the original ADM. In the ADM, the main assumption was to develop the Target Architecture on a seemingly fixed Baseline Architecture. This model reflects the reality that was discussed earlier in the chapter, namely that enterprises needs to manage both long term planning and short term reactions. There will never be a permanent baseline to build upon. Rather, a Target Architecture is to be accomplished, against a baseline that itself evolves. The baseline is constantly being modified through *solution architecture*. There is no commonly accepted way to define solution architecture, but it can be said, that enterprise architecture and solution architecture touches on virtually all the same subjects, but with different perspectives and different contexts. [Temnenco] Solution architecture is much more narrow in scope compared to the ADM, and also takes a more short term view. A typical example of solution architecture would be a business process improvement project, including creation or adaption of business services.

The second principle follows from the first; because the Engagement Model has been separated from the ADM cycle, the architecture development process is now also a separate entity. The architecture process consists of phase B-E in the ADM. The ADM is still guided by the Architecture Vision, which is rooted in strategy.

Finally, there has been no changes to the way strategy is used to drive the architecture effort. This means, that the Operating Model has been discarded as the primary tool for making decisions about architecture requirements. This is not to say that the Operating Model is not a valuable thinking tool; it is however likely to be more integrated with strategic considerations.

5.5 Challenges of Integrated Enterprise Modelling

Having seen the modified architecture development method, it will now be the time to look at some of the challenges of realising this model.

One critical point, which can be raised against the approach sketched above is that the architecture work becomes more complex. This is true; by pursuing both long-term and short-term development cycles at the same time, the architecture development process invariably becomes more complex. The counter-argument is however, that an approach that primarily emphasises long-term planning is an unrealistic simplification. In other words, separating the Engagement Model from the process of developing the architecture, may increase perceived complexity, but this increase is just a reflection of the real complexity in managing both long-term planning and short-term changes at the same time.

Managing this complexity is however a major challenge. To manage the complexity, it would be necessary to have visibility through the architecture artefacts. In EA, documentation in form of EA artefacts is typically used to reduce complexity. EA artefacts are highly abstract and codified representations of “physical” things in the architecture. Often the artefacts are modelled using DSML's, which encapsulates the domain knowledge. So having the individual components in the architecture documented will not be enough. Unless the enterprise can understand the relations between components in the enterprise with relative ease, then an inordinate amount of resources would have to be used to understand the impact of introducing changes in the architecture. Traceability thus becomes a critical capability; it should be possible to understand the enterprise architecture as a coherent system, instead of just as individual components. This sentiment is much in line with the IEEE 1471:2000 definition of architecture, which emphasises the relations between components.

The purpose of this part of the chapter is to understand the obstacles to such a language, but also to build a foundation for a solution.

5.5.1 Characteristics of Enterprise Modelling

The next step is to understand the nature of contemporary enterprise modelling. Enterprises are already having a variety of different modelling practices and standards. Different models makes up, what can be referred to as the *enterprise model set*, which is a “group of conceptual models built to obtain a coherent and comprehensive picture of an enterprise. [Dursum & Perakath:257] Models in the enterprise model set are usually sharing three critical characteristics: [Dursum & Perakath:259]

- **Different in nature:** All model types are different in nature from other types of models, i.e. the model captures information different from other models. The difference lies in the different semantic categories of primitives that the model builds upon (such as business processes, tasks, services etc.). This is not dissimilar to the Zachman Framework, where each cell describe a normalised aspect of the enterprise.
- **Equal importance:** All model types are equally important in making up a complete description of the enterprise.
- **Dependencies:** All models constituting the set are dependent rather than independent, as each model depends upon and is constrained by aspects captured in other models.

So the first part of explaining modelling heterogeneity is to understand that the many different modelling practices, and the many different modelling standards, each serves different purposes. Different stake-holders have different modelling needs, and each DSML helps to meet the specific needs of a stake-holder. But although each modelling paradigm uniquely fits the purpose of a stake-holder, the different modelling paradigms also turns models into silos that encapsulates domain specific information.

The second main reason for modelling heterogeneity is caused by immature modelling practices and immature modelling standards. It can be said that *“the current practice of enterprise architecture often comprise many heterogeneous models and other descriptions, with ill-defined or completely lacking relations, inconsistencies, and a general lack of coherence and vision”*. [Lankhorst et al.48] Although a rather bleak description, this observation is very much in line with the findings concerning the model-driven perspective; there are still considerable gaps in enterprise modelling.

TOGAF acknowledges that *“there is a need to provide an integration framework that sits above the individual architectures”*, and divides “integratability” into low-end, and high-end. [TOGAF:29] At the low-end, architecture descriptions have a look-and-feel that is *“sufficiently similar to enable critical relationships between the descriptions to be identified, thereby at least indicating the need for further investigation”*. [TOGAF:29] At the high-end integratability means that *“different descriptions should be capable of being combined into a single logical and physical representation”*. [TOGAF:29] The TOGAF finally goes on to conclude that: *“At the present time, the state of the art is such that architecture integration can be accomplished only at the lower end of the integratability spectrum”*. [TOGAF:29] If this conclusion stands to reason, then it would be extremely difficult to achieve an integrated approach.

5.5.2 Strategies for Information Integration

So how is it possible to bridge the small worlds (i.e. all the domain specific modelling paradigms) into a big world (i.e. consolidated) view of the enterprise? In general, when sharing information across domains, there are three available strategies: [Uschold, Jasper & Clark:2]

1. **Use of sharing services via point-to-point integration:** Two or more systems exchange information via run-time interactions.
2. **Neutral authoring:** A neutral language is used for authoring information
3. **Neutral interchange formats:** Knowledge is exchanged between systems, via an “neutral” intermediate format.

The first solution of using point-to-point integration services, leads to the problem of managing $n*(n-1)$ connections between the information sources.⁷⁶ Given the many different sources of information, this solution does not appear manageable. The second solution concerns the use of a neutral language for authoring; individual parts of the global model would then be transformed into domain specific models. This solution is very complex and does not seem attainable.⁷⁷ It is difficult to imagine a common language, which would cover all the needs that are met by DSML's. Moreover, end-users are often reluctant to modify existing modelling practices and working methods. [Terrasse et al.:21]

The final solution concerns the use of a neutral interchange format. Through the use of a common enterprise language, all stake-holders would be able to understand the relations between components in the architecture. Such a language would be easier to implement, as some degree of information loss can be accepted; the purpose is to make generalisations about the domain, rather than including all possible attributes. A simple and neutral language could in particular be approximated towards the minimum set of attributes, that could satisfy the need of understanding the enterprise as a whole.

5.5.3 The Enterprise Meta-model

The first step in creating a new language is to define the abstract syntax expressed through a meta-model. The purpose of this section is just that; to define a simple enterprise meta-model. Meta-models are important tools, because they help describe the enterprise in abstract terms, including the basic concepts that may appear in a concrete model, [Stahl & Völter:19], and the different types of relationships that can be drawn between those concepts.

⁷⁶ Assuming that connections are uni-directional

⁷⁷ Of course a language like the MOF could be used to construct such a language. But the authoring language would have to be built on to of MOF, and this language would in itself be unwieldy and unmanageable.

I should however point out, that the enterprise meta-model may sometimes be defined rather implicitly, as many EA repositories will have a pre-defined (and typically proprietary) meta-model. If an enterprise has such a repository, has all the relevant information stored in the repository and if the meta-model fits the needs, then meta-modelling (and meta-data integration) need not exist as an explicit discipline. Moreover, most EA literature seems to take for granted, that an EA repository exists in the enterprise, and that this repository is capable of holding and integrating all relevant information. As such, meta-modelling is not a particular well-described discipline in regards to EA. This I think, is a limitation of current EA literature. To enable more mature modelling practices, enterprises must understand the importance of the enterprise meta-model.

An important benefit of meta-models are that they raise the level of abstraction. By defining the concepts to be used in the enterprise meta-model on abstract level, the meta-model also becomes a vocabulary of these concepts. Moreover, defining connections between models on a micro level would be extremely resource intensive. Instead, a smaller set of types of relations can be defined on meta-level.

In the following, I will establish a meta-model, which will be used later in this thesis, as I will seek to implement the shared language. The overall design principle of this meta-model will be simplicity. There are two main reasons for this. The first reason is related to the purpose of including a meta-model in this thesis; the purpose is not to make a ready-to-use meta-model, but rather to create a proof-of-concept, which can be considered good enough to demonstrate the importance of such an approach. Secondly, from a usage point of view, it can also be argued, that attempting to merge all possible information sources and providing a cumulated view of all attributes, would leave too much complexity on the end-user. [Vdovjak & Houben:2] This would defeat the purpose of reducing complexity.

Besides the main principle of simplicity, the meta-model will be developed on the basis of two different use-cases:

- **Line-of-sight:** The meta-model should support line-of-sight, i.e. traceability across all architecture domains. Moreover, strategy concepts should be included as well, in order to provide traceability all the way from strategic initiatives and down to the technical architecture. The line-of-sight will be an important tool in being able to understand the architecture as a whole, and thus providing for impact analysis. The meta-model will thus include strategy, business architecture, application architecture, data architecture and technology architecture.
- **Versioning:** The meta-model should be constructed in a way that makes it possible to version components, as well version aggregate artefacts (such as on baseline architectures, business architecture etc.). As it was concluded in the previous chapter, many changes to an architecture will have to be deployed in unison due to coupling. If not, there will be a considerable risk of introducing negative side-effects by unitarily changing one component in the architecture.

The meta-model has to a large degree been defined by the use of TOGAF concepts. It has however been enriched with constructs that relates to BPM or SOA. The entire meta-model can be seen in figure 25. The figure can also be seen in full figure in appendix G.

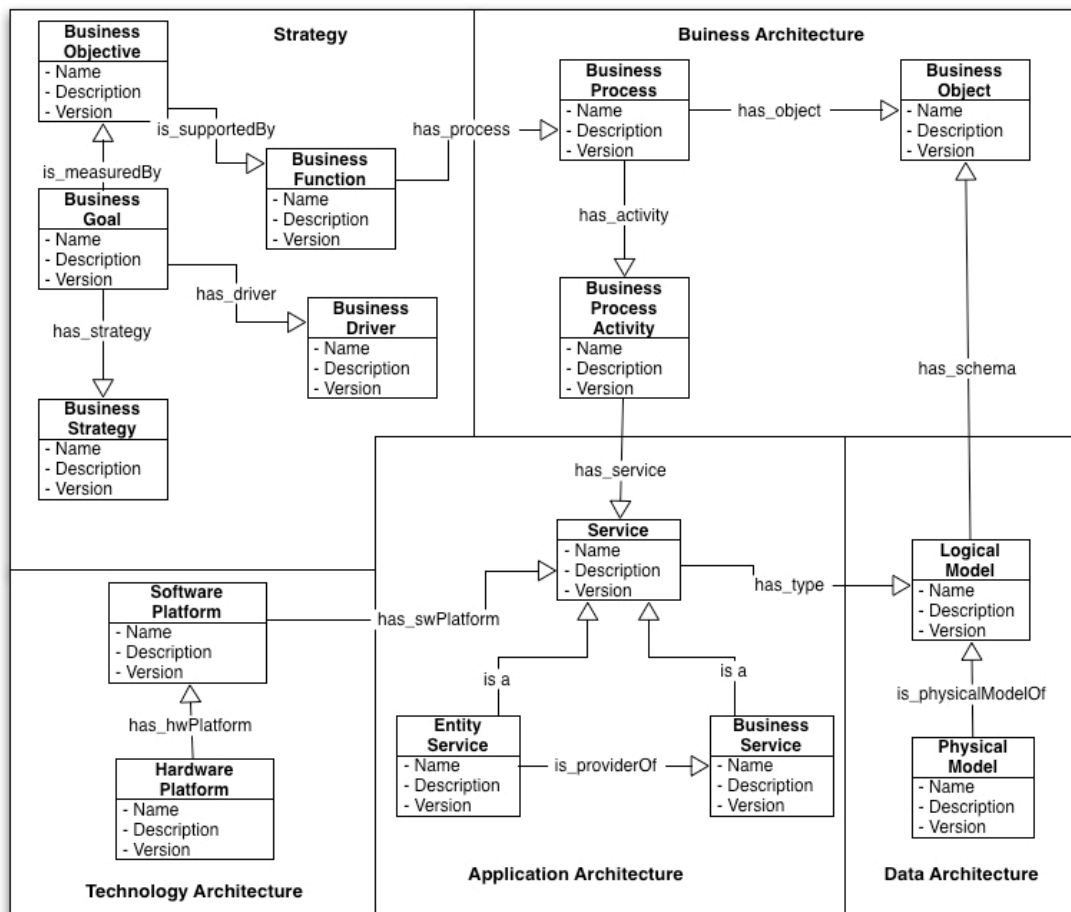


Figure 25: Meta-model [Own production]

For most parts, the meta-model is likely to be fairly obvious to the reader, given the ground that has been covered in the thesis so far. More documentation is however provided in the appendices:

- **Concept / Class Hierarchy:** Appendix H documents the five domains that makes up the meta-model. For each domain, the hierarchy of concepts are presented in a figure. Furthermore, each concept is described shortly.
- **Semantic connections:** The types of semantic connections that are included in the meta-model are documented in appendix I. The documentation includes a description of each type, as well as parent child relations to concepts.

There are however a few clarifications:

Firstly, as the ADM itself does not deal with SOA, there are no related artefacts defined by the ADM. Business services and entity services has however been added to the meta-model. Activities in business processes can consume services through use of the `has_service` object relation.

Secondly, it is important to make a remark concerning the modelling of business objects (or business concepts). The Data Architecture deals primarily with logical and physical modelling of data. The *Business Data Model* in contrast, is an output of the Business Architecture phase in the ADM. Little guidance is given as to the exact primitives in this data model, except that it concerns “*entities, attributes and relationships*” [TOGAF:61] It stands to reasons, that such a *Business Data Model* would have to include semantic definitions of business objects. At least, such an approach would make sense in the context of business modelling. Therefore, the concept of a business object has been added to the Business Architecture. The business object is expressed through one or more logical models (schemas).

Finally, the strategy domain is defined, based on the terminology used in TOGAF.

5.5.4 Requirements for a Concrete Syntax

The abstract syntax, expressed by the meta-model, will however be of little value, unless it can be implemented in one or more concrete syntax's. Without the implementation of a concrete syntax, the meta-model will just be a “pretty drawing”. There are some high-level requirements for the concrete syntax:

- **Machine-readable:** The first requirement is that the language should be implemented in a machine-readable standard format, i.e. in a format that can be persisted to files and then loaded again as needed.
- **Human-understandable:** The concrete syntax should also be available in a form that is understandable to humans. The purpose of the enterprise architecture language is exactly to facilitate analysis and communication about the architecture.
- **Support viewpoints:** Moreover, to support analysis and decision making it is vital that it is possible to provide stakeholders with viewpoints that support their particular need. A viewpoint is a “*selection or derivation*” of another model. [Lankhorst et al.:153] Few, if any stakeholders, will have interest in viewing a model of the entire enterprise. The viewpoints can be pre-defined, or defined ad-hoc to support specific analysis requirements.
- **Machine-understandable:** Finally, it would be preferable to have a language that is machine-understandable. Given the number of “moving parts” in a BPM-SOA based architecture, which is built for change, managing all the different parts manually becomes increasingly difficult. Thus, the enterprise language should support automated information gathering about the enterprise architecture. Digitally stored models are machine-readable, but not necessarily machine-understandable.

The purpose of the next chapter is to devise a way to implement the concrete syntax.

5.6 Chapter Summary

The purpose of this chapter was to gain an understanding of the role of EA in the context of BPM-SOA. The TOGAF ADM was used as basis for the analysis. The chapter consisted of five parts.

In the first part, I briefly set up the relationship between BPM-SOA and the ADM. Although the ADM is a generic approach to architecture development, certain adaptations have to be made to accommodate SOA. I referred to the SOA/TOGAF Practical Guide project for this. In particular, it was pointed out, that SOA services fit well into the Application Architecture. As for the role of BPM in the ADM, I stressed the striking similarities between the artefacts of BPM and the output from the Business Architecture phase. Moreover, both BPM and the ADM emphasises the notion of as-is and to-be modelling.

In the second part of the chapter, the relationship between the ADM and agility was established. I found that the ADM is a very good tool for creating both strategic fit and functional integration. In this way, the ADM provides a structured methodology to build flexible and sustainable architectures. On the other hand, the ADM methodology was not found to be particularly agile. Rather, the ADM seems to rest on the assumption, that the baseline is fairly stable, and that changes to the baseline can be treated as exceptions. Such an approach would not fit well with the intentions of BPM-SOA.

In the third part of the chapter, the ADM approach was contrasted to that of the Foundation for Execution. This model was found to hold some appealing aspects, especially in regards to the separate Engagement Model. It was also found, that the Operating Model provided important questions as to the scoping of BPM-SOA across business units. On the negative side however, the Foundation of Execution was found to be very IT-centric. Moreover, the approach did not contain tools or methodology for the EA work.

In the fourth part of the chapter, I integrated the knowledge from the ADM, with the knowledge gained from the Foundation of Execution. This knowledge was codified in a model. Most importantly, the architecture process was separated from the Engagement Model. This would have the consequence, that the architecture could be developed both through long-term thinking, but also through continuous iterations of the baseline by use of solution architecture.

In the final part of the chapter, the challenges of pursuing both long-term planning and short-term changes was discussed. This would necessarily make the architecture process more complex, but this complexity is a reflection of the underlying complexity of managing both long-term and short-term planning. I found that a common language for EA is needed, which could provide traceability across architecture artefacts. Such a language would include both an abstract syntax, expressed by the enterprise meta-model, as well as one or more concrete syntax's. The meta-model was defined, based on TOGAF concepts, enriched with BPM and SOA specific concepts. Moreover, two use-cases that could demonstrate feasibility was described. Finally, four requirements for an implementation of the language was defined.

Module 3 – Connecting the Dots

Chapter 6 – Implementing a Common Language

In the previous chapter it was argued, that a common language across the architecture artefacts was needed. The abstract syntax was defined by a meta-model. In this chapter, I will investigate the opportunities for establishing the concrete syntax to implement the language. In particular, I will look towards ontologies and ontology representation languages, to see which features they hold in relation to describing and exchanging meta-data.

In the first part of the chapter, I will look at ontologies from a conceptual level. Initially, a definition and a short description of ontologies will be given, as familiarity with this field is not expected from the reader. From there, I will argue that there are certain characteristics of ontologies, that makes them ideally suited for representing and sharing knowledge within a domain. In this context, key objectives will be to understand how ontologies relates to meta-models and how ontologies differs from taxonomies.

In the second part of the chapter, I will analyse how ontology representation languages can provide a concrete syntax for the common EA language. The search for such a representation language, will take its departure in Semantic Web technologies. The purpose will be to investigate whether ontology representation languages can be used to implement the concrete syntax. In particular, the implementation should satisfy the requirements set forth in the final part of last chapter, i.e.:

- Machine-readable
- Human-understandable
- Support viewpoints
- Machine-understandable

6.1 Ontologies

The purpose of this part of the chapter is to understand ontologies on a conceptual level, Among the key discussion points for this part of the chapter will be to understand how ontologies relates to other important concepts, such as meta-models and taxonomies.

6.1.1 What are Ontologies?

Ontology is a term borrowed from philosophy, which refers to the science of describing the kinds of entities in the world and how they are related. [OWL] Ontologies can also be considered “a *systematic account of existence*”. [Gruber:1999] The main feature of ontologies are that they can “*provide a shared and common understanding of a domain that can be communicated across people and application systems*” [OWL] In the simplest form, ontologies can be defined as: [Gruber:1999]

“a specification of a conceptualization”

A conceptualisation refers to an abstract, simplified view of a problem domain. The specification means, that the conceptualisation should be done in a formal, and declarative way. [Gasevic, Djuric & Devedzic:46] Ontologies are used for making clear and precise statements about the world. From a theoretical point of view, ontologies are high level models that describes the problem domain, completely independent of implementation details. Thus, the ontological model offers a huge reduction of complexity; by making generalisations about the problem domain, the domain becomes easier to understand and manage. It goes without saying, that the ability to reduce complexity is especially interesting in the context of being able to create an understanding of complex architectures.

Hendler provides a more elaborate definition of ontologies:

“a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of inference and logic for some particular topic” [Hendler]

Already at this point it becomes evident, that there is a huge conceptual overlap between ontologies and meta-models. They are both conceptual models of a problem domain, and just as an ontology specifies the constructs and rules to represent the domain, so does the meta-model. Meta-models can be said to be *“ontologies used by modellers”*. [Gasevic, Djuric & Devedzic:73] Roughly speaking, Hendlers definition can be separated into three parts; the vocabulary, the semantic interconnections, and the rules of inference and logic.

The first part concerns the vocabulary, which is a list of abstract concepts that makes up the domain (enterprise), such as *services*, *business processes* etc. This is similar to the notion of a vocabulary in meta-modelling.

The second part concerns the *semantic interconnections*. There are two important observations to make in this regard. Firstly, the notion of *semantic interconnections* implies, that the meaning of the connections between the concepts should be defined in the ontology. This guides us towards semantics that is not only machine-readable, but also machine-understandable.⁷⁸ Secondly, the notion of interconnections is a key difference between ontologies and taxonomies. Whereas taxonomies are hierarchical classification schemes, used for organising the concepts in the vocabulary, ontologies are also used to draw the relationships between instances of these concepts. Taxonomies and ontologies are however closely related. An ontology can be thought of as consisting of two parts, which can be seen depicted in figure 26.

⁷⁸ The notion of machine-understandable semantics will become much more apparent during the discussion of ontology representation languages.

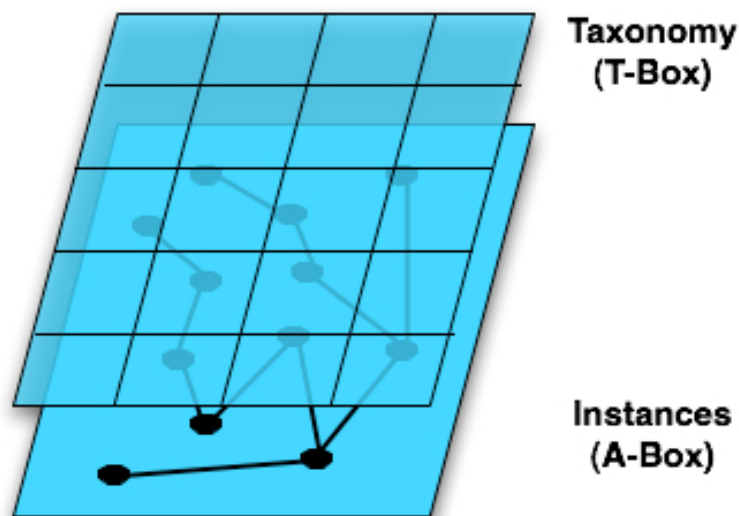


Figure 26: Ontology Layers [Own production]

The upper layer depicts of the “*terminological box*” (T-box in short). The T-box describes the class (or concept) hierarchy. As the T-box orders classes into a hierarchy, the T-box can in fact be considered a taxonomy. In contrast, the “*assertional box*” (A-box) concerns knowledge about specific instances and their properties. There are two main types of properties; object properties defines relations between instances and data properties associates constants to instances.⁷⁹ So the A-Box provides the main distinction between taxonomies and ontologies. Distinguishing between taxonomies and ontologies in relation to enterprise architecture is important. A primary focus in EA has been the establishment of taxonomies, such as the Zachman Framework or the Enterprise Continuum in TOGAF. As the ontology describes instances and their relations, the ontology can become an important addition to the taxonomy.

The final part of the definition concerns rules of inference and logic. A main reason for striving towards machine-understandable semantics is the possibility to make inferences, i.e. the ability to find or infer information based on the information in the ontology. [McComb] Inference is distinct from queries, because it allows returning answers, that are based on information not explicitly stated. [Parreiras, Staab & Winter:5] Queries only returns answers based on explicitly stated information. As will become apparent later in this chapter, ontology representation languages plays a crucial role in enabling inference.

⁷⁹ A third property type is the annotation property

6.2 Ontology Representation

Using ontologies for knowledge sharing, requires formalism for expressing them. Informal ontologies, such as those expressed in a series of natural language statements or as informal visual drawings, relies too much on the user-agent for interpretation. This is especially a challenge in regards to making the meta-data machine-understandable. Rather, it should be possible to make á priori assumptions about how data is to be interpreted; a more formal data model is needed. [Gasevic, Djuric & Devedzic:83].

The role of ontology representation languages is to provide the formalism to describe instances of ontologies and to encode the ontologies themselves. But several requirements needs to be balanced when selecting (or designing) an ontology representation language. An important design decision regarding ontology representation languages concerns the level of expressiveness, which refers to the richness with which the problem domain can be expressed. The expressiveness is decided by the different axioms that can be used in the language. Axioms are constructs that are used to “*to provide information about classes and properties*”. [OWL]

Ontology representation languages do however also need to balance the need for expressiveness against other factors, such as the need for computational completeness⁸⁰ and decidability.⁸¹ A final factor concerns the possibility of implementing the software programs and reasoning algorithms needed for automatic reasoning; the higher degree of expressiveness, the harder it is to implement inference engines to support the reasoning.

82

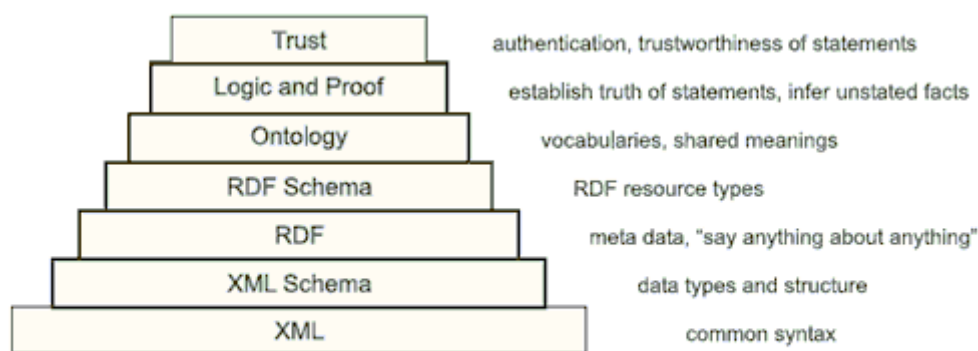
80 Computational completeness, i.e. that all entailments of the ontology is calculates

81 Decidability, i.e. the reasoning about the ontology should be done within reasonable amount of time.

82 An inference engine is a software program that infers knowledge from a knowledge-base, i.e. an ontology.

6.2.1 Ontologies and the Semantic Web

Despite its huge success and influence on society, the current incarnation of the internet is not without its limitations. While the content on the web is machine-readable, little of it presented in a machine-understandable way. Machines can parse basic web site elements, such as the title, meta-tags links etc. Most content is however stored in unstructured natural language documents. The Semantic Web envisioned by Tim Berners-Lee is an attempt at addressing some of the shortcomings, by making the content more understandable for machines [Heflin & Hendler:54] The Semantic Web is interesting in the context of this thesis, because it deals with the description and interexchange of meta-data in a way that can be understood by both humans and machines. Important concepts of the Semantic Web can be seen depicted in figure 27.



(adapted with changes from Berners-Lee,
<http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html>)

Figure 27: Semantic Web Layers [Passin:14]

The two upper layers are out of scope for this thesis. The two lower levels contains XML and XML Schema. Neither of these two languages deals explicitly with semantics, and Both languages relies on external documentation for semantic descriptions, such as annotations, links or hooks to external documents.⁸³ Modern ontology representation languages are usually built on XML, and XML Schema, which provides an unambiguous way to encode syntax.

⁸³ XML represents data through an object-tree structure, and the use of ID/IDREF mechanisms. As such, it does provide some semantics. But since XML does not provide much insights into the interpretation of data, it does not help us represent ontologies in way that can be understood by machines.

6.2.1.1 Resource Description Framework (RDF)

Moving up the stack, the Resource Description Framework (RDF) is found, which “*allows the specification of the semantics of data based on XML in a standardized interoperable manner*”. [Gómez-Pérez & Corcho:55] In other words, RDF provides a formal language for describing and exchanging information about resources. Statements about resources are expressed through triples, which follows to the subject-predicate-object pattern.⁸⁴ The triple pattern can be considered the abstract syntax of RDF. Figure 28 depicts a triple, illustrated by use of a *semantic network*.⁸⁵ Semantic networks uses nodes, arcs and squares as primitives, where arcs between nodes defines (unidirect) arbitrary relations. Literals are illustrated by use of squares. The semantic networks and RDF shares data-model formalism. [Gómez-Pérez & Corcho:55] Semantic networks can thus be regarded a concrete syntax for RDF.

The triple illustrated in the figure, describes the natural language sentence: “the ball is yellow”. A triple consists of a subject, a predicate, and an object. The subject identifies what the triple is describing, i.e. in this case the ball. The predicate defines the property or characteristics about the subject that we are describing. The object is the actual value, i.e. in this case yellow.

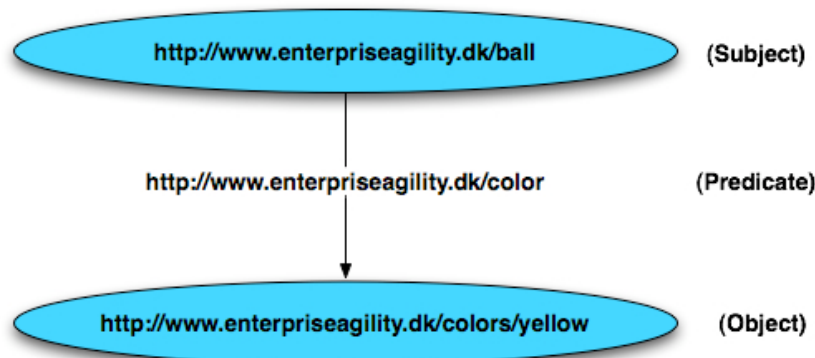


Figure 28: Example of RDF Triple [Own production]

⁸⁴ The pattern is sometimes also referred to as the Object-Attribute-Value pattern (O-A-V pattern). See f.ex. [Gasevic, Djuric & Devedzic:15]

⁸⁵ It should be noted that there are several variations of semantic networks. A more thorough investigation of semantic network types is given by [SOWA]

A more complex example is depicted in figure 29. There are several interesting features in this example. Firstly, RDF can include objects that holds references to literals. The literal *title* is denoted by the square: the subject *thesis* has a *title* called *Enterprise Agility*. Secondly, it is possible to mix subjects, objects and predicates from different sources. In the example, the predicate *student* refers to a (fictional) *studentid* at www.itu.dk, with the *studentid* equal to 12345. Because of this ability to mix sources, and because URIs are used to point to other resources, RDF is essentially a distributed format by design. The *programme* predicate further tells us, that the student is enrolled at Ebuss programme. The last feature requires a bit more explanation; because an object can hold references to another resource, it is possible to build graphs of interconnected resources. An RDF graph is simply a set of RDF triples. A sub-graph of an RDF graph is a subset of the triples in the graph. The graph expressed in a set of triples can be seen below:

Subject:	Predicate:	Object
www.enterpriseagility.dk/thesis	www.enterpriseagility.dk/title	Enterprise Agility
www.enterpriseagility.dk/thesis	www.itu.dk/student	www.itu.dk/studentid/12345
www.itu.dk/studentid/12345	www.itu.dk/programme	www.itu.dk/programmes/ebuss

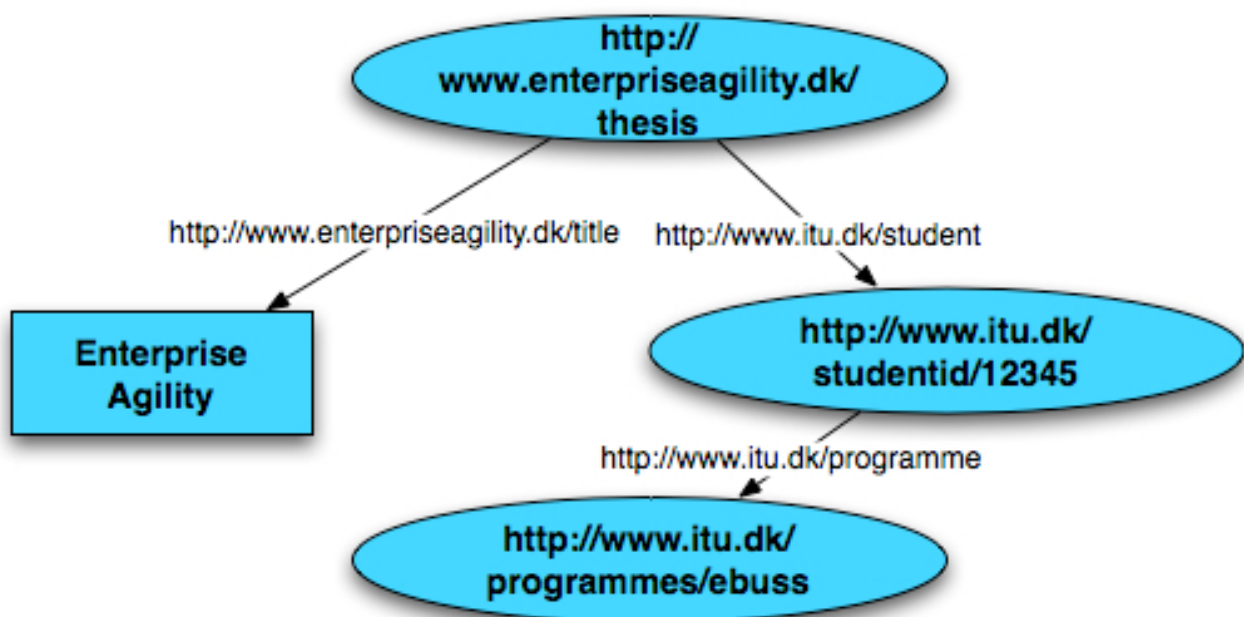


Figure 29: Complex RDF Example [Own production]

RDF can also be persisted to files in standardised file formats. There are several different syntactic formats available. Examples are RDF/XML, Notation 3 (N3), and N-Triple. The current W3C adopted standard is RDF/XML. N3 is a rather popular standard for encoding RDF in XML , which holds the distinct advantage over RDF/XML, that it encodes triples in a more tabular way. This provides a more uniform representation than that of RDF/XML. N3 is also easier readable by humans than the XML counterpart. Finally, triples can also be persisted to triple-stores, which can be built-for-purpose databases or regular relational databases with an RDF layer on top.⁸⁶

It is essential to stress, that RDF is only used to describe instances of an ontology, thus only providing a domain-neutral mechanism to describe resources in a domain. RDF does not by itself contain any ontological constructs, which could be used to make á priori assumptions about a domain.

⁸⁶ On [ESW] a list of triple stores that are scalable to a large number of triples is provided. Incidentally, there is little information as to the response times of said triple stores under maximum load, which makes the scalability claims rather dubious for practical purposes.

6.2.1.2 RDF Vocabulary Description Language (RDFS)

The first level of ontological constructs (axioms) is defined by the Resource Definition Framework Schema (RDFS). RDFS extends RDF by providing a vocabulary of frame-based primitives, much similar to classes in object-oriented programming. This allows for the creation of stereotypical representations of concepts. RDF Schema represents a standard way to encode ontologies, providing a way to make *á priori* assumptions about the semantics of a problem domain. RDFS also uses RDF to describe the RDF vocabularies. This makes RDFS self-containing, as meta-data and data are described together and with the same syntax

The most prominent primitives of RDFS are *Class*, *SubClassOf*, *Property*, and *SubPropertyOf*. The *SubClassOf* primitive is the basic construct for building the taxonomy.⁸⁷ [Gómez-Pérez & Corcho:57] Properties can be considered “*attributes of resources and in this sense correspond to traditional attribute-value pairs*”. [RDFS] Properties are defined independently of classes. This means that a property can in theory be applied to any class. The RDFS constructs *domain*⁸⁸ and *range*⁸⁹ can however be used to restrict the use of properties. In this way, they can indirectly be attached to certain classes. [OWL]

It should be noted, that RDFS is a rather limited ontology representation language, as it only contains few ontological primitives. Therefore, it only provides little opportunities for reasoning about a problem domain. Most importantly however, RDFS allows sub-classing and the definition of properties.

⁸⁷ In [Gómez-Pérez & Corcho:57] the authors further describes three other primitives for defining taxonomies in ontology representation languages. Neither of these three are implemented in RDFS.

⁸⁸ The `rdfs:domain` property is used to state that any resource that has a given property is an instance of one or more classes.

⁸⁹ The `rdfs:range` property is used to state that the values of a property are instances of one or more classes

6.2.1.3 Simple Protocol and RDF Query Language (SPARQL)

As the name suggests, SPARQL is both a protocol and a query language. As a query language, SPARQL can be used to perform queries across various data sources, “*whether the data is stored natively as RDF or viewed as RDF via middleware*”.⁹⁰ [SPARQL] The protocol specification of SPARQL concerns remote invocation of SPARQL queries over either HTTP or SOAP. The most common use of the SPARQL term refers to the query language. It is in similar vein, that I will use the term in the rest of this thesis.

The purpose of SPARQL is to provide a human friendly query-syntax for application developers. In this sense, SPARQL is much similar to SQL. This is also evident in the choice of language constructs, which includes SELECT, WHERE and ORDER BY clauses. Most queries in SPARQL follows a *basic graph pattern*, which is similar to a regular triple, except that “*each of the subject, predicate and object may be a variable*”. [SPARQL] There are two basic ways to return results from SPARQL queries; SELECT queries returns result sets, which are tabular representations of query answers, encoded in the SPARQL Query Results XML Format. The CONSTRUCT query in contrast, returns the result as RDF graphs. Furthermore, CONSTRUCT can be used to merge several graphs or retrieve sub-graphs. The ability to merge, extract and export graphs are very interesting features of SPARQL, especially in regards to being able to create different viewpoints.

Finally, because ontologies expressed in RDFS are self-contained, it is also possible to query their meta-data via SPARQL. This is unlike f. ex. in a relational data system, where meta-data queries would involve querying system tables. [Baurmann] This means, that is possible to use SPARQL to “*find values for partially known graph structures*”, [Beckett:10] as well as for getting “*information about an identifiable object with unknown properties*” [Beckett:10] In the relational world, the entire RDBMS platform would be a self-containing system. This makes the ontology an abstract model of a domain, that can be moved freely around between different applications. [Baurmann]

⁹⁰ An example is the D2R Server and the D2RQ mapping language (that describes mappings between relational data and ontologies), which provides the opportunity for treating relational data as RDF graphs, including querying relational data via SPARQL. Moreover, the DARQ project attempts to establish a standard way to perform federate queries over multiple data sources.

6.3 Using RDFS to Implement the Meta-model

In the previous chapter, four requirements for a the concrete syntax's of the common language were laid out. The purpose of this section is to conclude whether RDFS actually meets these four demands.

The first requirement was that the language should be machine-readable, i.e. it should be possible to load and save models expressed in the language, through a standardised file format. It was found that this is the case, although there are several syntactical versions available. The serialisation formats are generic, i.e. they do provide a standard way to express facts about resources by using triples, as well as a standard way to express the data-model. Because data- and meta-data are stored together, and with the same syntax, no external schema is needed for understanding the semantics of the ontology.⁹¹

The second requirement was that the implementation should contain a concrete syntax, suitable for human interpretation. Semantic networks provides such a notation, as semantic networks shares data-model with RDF. Moreover, semantic networks naturally offers the same reduction of complexity as the use of RDF triples does. Semantic networks are however a generic way to express the domain. More visually appealing graphical notations (including intuitive symbols mapped to concepts in the vocabulary) could be developed on top of the ontology.⁹²

The language implementation should also support the creation of viewpoints. Through SPARQL, which is an application-developer “friendly” query language, it is possible to query the ontology, as well as merging or extracting graphs.

⁹¹ However, not all RDF triples in the file are necessarily described by the data-model. This is entirely up to the ontology designer.

⁹² ArchiMate is f. ex. an example of a concrete syntax for service-oriented enterprise architecture. [Lankhorst]

The final requirement concerns the possibility to perform automatic reasoning on the model. RDFS provides some ontological constructs, which can be used to make simple *á priori* assumptions about the problem domain. Support for automated reasoning is however rather limited. A better candidate language for automatic reasoning would be the Web Ontology Language (OWL). In general, OWL comes with much richer facilities for expressing the domain, as many more axioms exist. OWL comes in three different dialects, each with a varying number of axioms included. The degree of richness affects the degree of computability and implementability of the language dialects. The three OWL dialects are: [OWL]

- **OWL Lite:** supports those users primarily needing a classification hierarchy and simple constraint feature. An example of this simplification is that OWL Lite supports only binary cardinalities
- **OWL DL:** supports those users who want the maximum expressiveness without losing computational completeness. Due to its expressiveness, OWL DL is harder to implement than OWL Lite
- **OWL Full:** meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. The expressiveness and the syntactic freedom means that OWL Full is not likely to be implemented in a way that fully support automatic reasoning.

There is no doubt that OWL-based ontologies will represent a huge step forward in reasoning capabilities. There is however at this point one major problem with OWL; there is no standardised equivalent to the SPARQL language. Since OWL is built on RDFS (with some limitations), SPARQL can be used to query OWL ontologies, but SPARQL does not understand OWL semantics. Therefore, I will go with RDFS for my implementation of an ontology. Moreover, since an RDFS-based ontology is simpler to implement than an OWL ontology, a likely strategy by ontology adopters will be to use RDFS as “bootstrapping”, later progressing to one of the OWL dialects.

So with some caution, it can be concluded that RDFS can be used to implement the meta-model, as designated in the previous chapter.

6.4 Chapter Summary

The purpose of this chapter was to investigate how ontologies and ontology representation languages could provide the basis for implementing the meta-model.

In the first part of the chapter, I investigated ontologies from a conceptual point of view. Ontologies are used to make generalisations about a problem domain, thus reducing the complexity in understanding it. Ontologies are much similar to meta-models, as they are both conceptual models of a domain, including a vocabulary of the abstract concepts in the domain, as well as types of relations to be drawn between instances of concepts. Taxonomies and ontologies are very closely related, but ontologies extends taxonomies by also including instances and the relations between the instances.

I then proceeded to investigating the role of ontology representation languages. Such languages, are used to provide the formal means to describe instances of ontologies and to encode ontologies. I especially investigated the two Semantic Web related languages, RDF and RDFS. RDF provides for the description of instances of ontologies, whereas RDFS extends RDF with ontological constructs, that can be used to make á priori assumptions about a domain. Moreover, I also looked at SPARQL, which is a protocol and a query language at the same time. As a query language, SPARQL holds some important features, as it is possible to extract, merge and export graphs. This allows for the creation of viewpoints, and for exporting parts of the ontology for specific purposes.

I finally concluded, that RDFS could provide the means for implementing the meta-model, according to the four requirements. RDFS could definitely provide a serialisation format. a human understandable notation and the possibility to create viewpoints. Less convincingly, the RDFS language provides some basic reasoning facilities. I see OWL as a huge step forward in reasoning capabilities, but there is of writing this not yet a standard language for OWL, that allows for returning results as graphs, while also respecting OWL semantics.

Chapter 7 – The Project Scenario

The purpose of this chapter is to codify the main findings of this thesis through a case scenario. Moreover, I will demonstrate the implemented ontology, which defines the shared language. The ontology will be demonstrated through the two use-cases defined in chapter 5.

The chapter will be divided into three main parts. I will start by presenting the scenario description, which will provide background information, as well as highlight important concepts from the thesis. In the second part of the chapter, I will demonstrate an ontology, which serves as the concrete syntax for the enterprise architecture language. The ontology will be an implementation of the meta-model defined in chapter 5. Moreover, it will be based on RDFS as described in the previous chapter. I will demonstrate how the ontologies can be used to fulfil the two use-cases from chapter 5. In the final part of the chapter, I will reflect upon the need for a common enterprise architecture language in general, and the use of Semantic Web technologies to implement the language in particular.

7.1 Project Scenario – T.A.X

T.A.X. is a government agency, administrating tax laws at state level, including income taxes, car registration taxes, value added taxes and import/export taxes. The T.A.X. enterprise operates within a very complex field, including a complex body of laws, and with many stake-holders having different interests. Furthermore, T.A.X. has been subject to many structural reforms, which includes having to take on tasks previously being performed by local government institutions at county and municipality level. The merging of entities into T.A.X. was often done in a haphazard way, due to financial and time constraints. The negative effects had become apparent both on the business and the IT side. On the business side, there were concerns that managers had to little visibility into the business processes, and thus had difficulties managing them.

On the IT-side, the many mergers of smaller units into T.A.X. had meant, that a large portfolio of legacy systems had been accumulated. The integration of these systems had turned out to be very complex and brittle. Moreover, massive duplication of logic exists within the architecture. In short, the architecture of T.A.X. can neither be considered efficient nor flexible. This makes it difficult to adapt business processes to new requirements.

To add injury to the insult, T.A.X. is increasingly facing new political demands. In recent years, the legislative pace has increased considerably, prompting the demand for implementing new tax regulations in shorter time. Moreover, the government sector at large is being pushed towards a higher degree of efficiency. *“Doing more with less”* is the new mantra. This has lead to a focus on digitisation in the public sector, which should increase the use of self-service, with the dual aim of becoming more efficient through automation, and providing a higher degree of customer satisfaction. Government enterprises maintaining their own IT-infrastructure would also become mandated to perform benchmarking against private market players. The thinly veiled threat is that poorly performing IT-functions will be outsourced or centralised to larger IT-functions.

In many ways, the current situation represents a “worst case scenario” for T.A.X. At political level, it was however recognised, that the current architecture of T.A.X. would be inadequate for meeting future demands. T.A.X. thus was provided funding, which was earmarked at performing a large scale modernisation of its architecture. In return for the provided funding, T.A.X. should ensure that the architecture would become more modular, ultimately ensuring better flexibility and improved opportunities for outsourcing.

Following the provision of funds for the modernisation program, the management of T.A.X. quickly came to the realisation, that they needed a structured methodology for this massive undertaking. They knew that they would need to balance both the long term modernisation of the architecture, with the ongoing changes to the architecture. As the use of TOGAF was well established within the government environment, using the ADM was the obvious choice.

The first iteration of the ADM would need to balance the two most pressing architectural needs:

- Improve process modelling and process management tools
- Improve service-oriented infrastructure

T.A.X. already had an established business modelling practice, but the diagrams mostly ended up as “shelf-ware” with limited usability. Thus, a priority of the business architecture development phase was to establish more efficient modelling practices in the form of a process architecture, including:

- **Process guidelines:** Standards, methods, guidelines, policies and tool selection. Modelling paradigms are chosen based on the existing practice in the enterprise, whenever applicable.
- **Process Models:** To provide a starting point for managing business processes a high-level overview of the processes in the enterprise was to be made. This would include visual representations of high-level processes, links between the processes and a list of end-to-end processes.

On the IT-architecture side, management quickly came to see SOA as a possible solution. At the onset of the modernisation program, the IT-function in T.A.X. had already been using SOA on a small scale to solve local integration issues. During the Architectural Vision phase of the ADM, the business case for SOA was developed and eventually approved.

The primary drivers behind adopting SOA was:

- A more flexible architecture due to loose coupling
- Ability to align service development projects with business requirements
- Increased modularity (as per the requirement for funding)
- Improve opportunities for re-use. This is especially relevant in regards to the different customer-facing web-portals.
- The ability to use business services as an abstraction layer between business processes and the underlying IT-systems. This would later provide for easier migration from legacy systems to new platforms.

Some drivers were considered for the business case, but was for various reasons de-emphasised or postponed to later:

- Opportunities for Business Activity Monitoring (BAM) would not be pursued until T.A.X. has a firmer grip on aligning enterprise goals/objectives with process design
- The potential to position T.A.X. as provider of shared business services to partners (either within the federated environment or outside) would also be postponed. The management of T.A.X. found that there were no obvious business model yet, and that T.A.X. would already have their hands full.

The experience with SOA in the IT-function had mostly been good, but it was also apparent, that a much more thorough approach was needed for large scale adoption of SOA. Moreover, some of the architecture development processes devised by the ADM, was deemed too generic. Therefore a process for service identification and specification was devised. The process was based on business process decomposition, and would yield a service realisation contract, including the technical interface for the services. Service development could then be transferred to the IT-function or to an external vendor.

It was realised, that governance would be vital in managing the effort. An Engagement Model was set up, including Migration Planning, Implementation Governance, and Architecture Change Governance. The Engagement Model would be responsible for linking overall architecture development with local projects.

7.2 The Common Language

The Engagement Model defines important governance processes, principles and incentives. As such, the Engagement Model is completely crucial in order to establish and maintain a sustainable architecture. To exercise the functions of the Engagement Model, it is however necessary to have full visibility through the architecture artefacts. As the architecture gets decomposed into individual components, each having their own life-cycle, the ability for any human to comprehend the entire architecture would get lost. So unless supported by IT, managing the many different components and their relations would be impossible.

The components and relations in an architecture can be understood at three different levels of abstraction. At the level of taxonomies, different types of concepts are categorised according to a hierarchical classification scheme. The second level is provided by the enterprise meta-model, which offers an abstract view of the concepts in the enterprise, including the types of relations that can be drawn between them. Finally, at the concrete level, instances of primitives will be defined and relations will be drawn between them. At this level, a common language is needed in order to provide a concrete syntax, i.e. a notation that can be understood by humans and/or machines.

7.2.1 Technology Choice

The concrete syntax for the common language can either be defined explicitly, f. ex. by use of ontologies or through the use of integrated repository solutions. Such integrated solutions may have more or less explicitly defined meta-models. The benefits and disadvantages of each of these two solutions must be weighed against each other. There is no inherently right solution. In the case of T.A.X. the choice was pretty clear. Faced with the choice of ripping out existing modelling practices, tools and repositories to acquire an integrated solution, the solution based on using RDFS ontologies for integration of meta-data seemed more appealing. It would not be a perfect solution, but the solution was deemed “good enough”. On the negative side, T.A.X. would be very much on its own. There would be no off-the-shelf solutions.

In the end the main drivers behind the ontology based approach was:

- **Flexibility:** The RDFS data-model was found to be extremely versatile. The ontology could easily be extended as needed
- **Based on open standards:** Basing the solution on an open standards would eliminate vendor lock-in. Moreover, open standards would allow for interoperability between the ontology and existing tools.
- **Cost:** The ontology could be developed and set up with very little capital expenditure.⁹³ Open source tools could be used in the mix.
- **Semantics:** RDFS is a natural language for describing semantics. This could potentially provide reasoning capabilities. Moreover, because data and meta-data is stored in the ontology language with the same formalism, it would be possible to browse the ontology without advance knowledge of the data-model.

Viewpoints: Finally, using the SPARQL query language it would be possible to perform queries or extract parts of the ontology. Exported graph could f. ex. be used as documentation, which could be used by vendors.

⁹³ The FEA-RMO is an ontology for representing the Federal Enterprise Architecture Reference Model. The decision to use an ontology for expressing the reference model was based on the principle of parsimony, i.e. an extreme unwillingness to spend money. [FEA-RMO]

7.2.2 Ontology Development

Before going on to demonstrate the ontology, I will give a brief description of how the ontology was developed.

The Protégé ontology editor was selected as a front-end for developing and interacting with the ontology. Protégé is a free and open-source program that “*support the creation, visualization, and manipulation of ontologies in various representation formats*”. The program is developed by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. Moreover, Protégé is rather extensible through its plug-in architecture. The Protégé editor also supports SPARQL in current versions.⁹⁴

The ontology has been created on basis of draft TOGAF 8 ontology, [OpenGroup] which has been modified in two different ways. Firstly, the ontology was converted into an OWL/RDF file from its original proprietary Protégé project format.⁹⁵ It should be noted, that although the ontology is stored in OWL/RDF format, only RDFS semantics will be demonstrated in the use-cases. Secondly, work was done in order to modify the ontology to fit with the meta-model described in chapter 5. Appendix H and appendix I details the concepts and the relations that was added. All visual notation diagrams has been made with the RDF-Gravity application, which is a free (but not open source) graph visualisation tool.⁹⁶ The map legend of the notation can be seen in appendix J. Finally, it should be noted, that object properties (relations) has been restricted by use of *rdfs:domain* and *rdfs:range* constructs, as properties are defined independently of classes.

⁹⁴ The version used is 3.4 (Beta). It appears that support for SPARQL will disappear form version 4.0 (which is currently in Alpha). Protégé 4.0 internally relies on a native OWL format, instead of the OWL/RDF format in the version 3.x series.

⁹⁵ An issue occurred in the conversion process. The original file contained some proprietary “Protégé Axiom Language” (PAL) constructs that were not compatible with OWL. Enabling PalConstraintsTab and trying to remove the PAL constraints did not work, as phantom PAL-CONSTRAINT entries were still in the OWL file. Removing the entries manually from the file solved the problem

⁹⁶ RDF-Gravity does however support RDQL queries, instead of SPARQL queries. RDQL predates SPARQL and only provides a limited subset of SPARQL. RDQL does f. ex. not support data-type checking.

7.2.3 Use-Case Demonstration

The use-cases will be demonstrated through a simple business process. The purpose of the business process is to register *VAT Returns Registration* documents. The business process is very simple, and only contains three activities i.e. a register activity, a control activity and a release activity.⁹⁷ The process can be seen in figure 30.

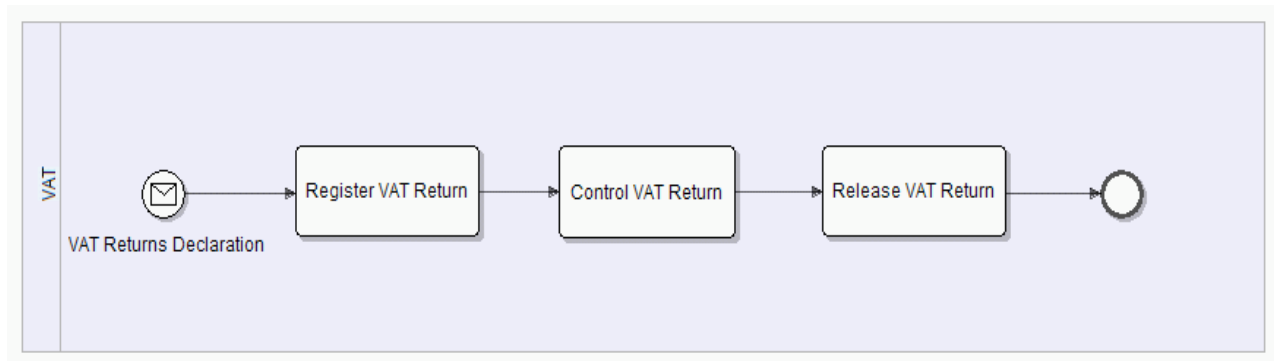


Figure 30: Example Business Process [Own production]

There is a single business object associated with the business process, i.e. the *VAT Returns Declaration* business object. The document is received in electronic form by means of online self-service, with the *VAT Returns Declaration* schema providing the logical structure of the interface data-type.

7.2.3.1 Line of Sight

The first use-case concerns demonstration of line-of-sight, i.e. traceability across all four architectural domains, as well tracing environmental links (i.e. links to the strategy domain). The central tenet of line-of-sight is that *from any given node, it should be possible to explore its relations to other nodes*. Such a tenet would imply, that data can be browsed without any advance knowledge of the data-model. This would f. ex. contrasts that of the relational database model, in which one would need to know the right tables to perform a query. Line-of-sight will especially be important in the context of impact analysis.

⁹⁷ The process and related meta-data has been kept to an absolutely minimum for pedagogical reasons. Thus, no exceptions, roles or decisions has been defined. The purpose is to provide just enough realism to demonstrate the use cases.

Thus, key competence questions to be answered could be:

- What is the impact of de-commissioning system x?
- What is the impact of modifying service y?
- What happens if the semantics of business object z is changed?

It should be noted, that a point of departure of these questions is that at least one node in the graph is known. The ontology would clearly be of help in answering the competence questions, because the ontology provides generalisations about the architecture, which in turn makes it easier to understand how things are interrelated. The ontology does however not (nor does it attempt to) replace the need for human cognition in performing the impact analysis. The ontology (as it is designed) describes structure, rather than behaviour. Figure 31 provides a simple overview of the business process. At the top of the figure, the type indicator for the process can be seen, which shows how data and meta-data can be accessed at the same time. Below the *VAT Returns Registration* process, the three activities have been associated, through the *has_activity* object property.

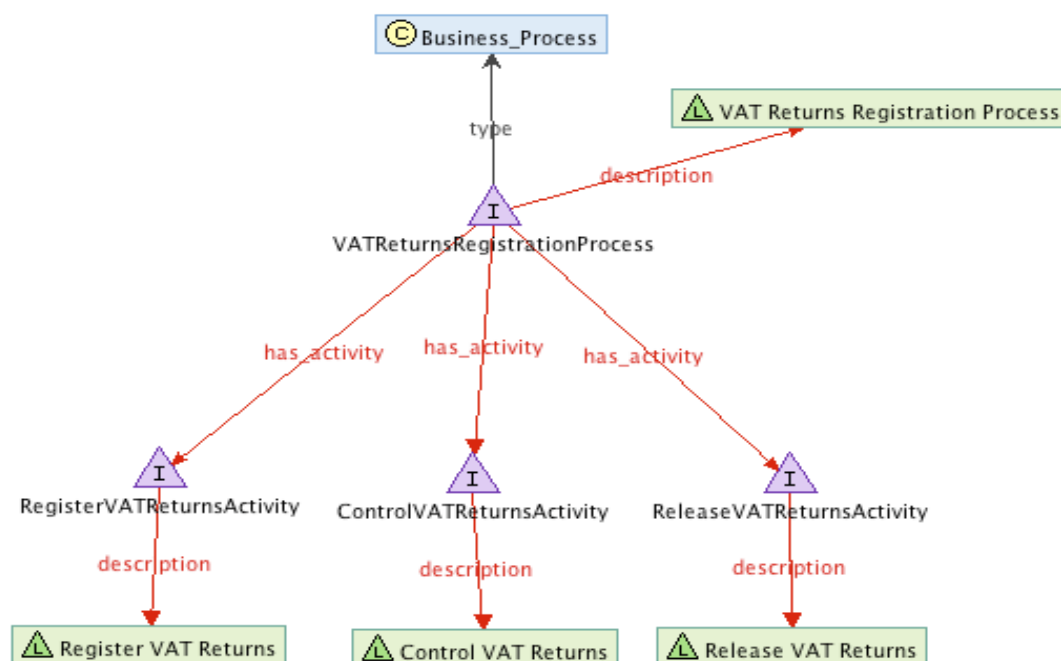


Figure 31: Simple Process Overview [Own production]

The figure does however not provide any information that could not have been gathered by a glance at the business process diagram. Thus, the next step is to drill down through the ontology. Rather, than trying to show all meta-data related to the business process, only meta-data relevant to the *Register VAT Returns* activity will be shown. Figure 32 below shows this selection.

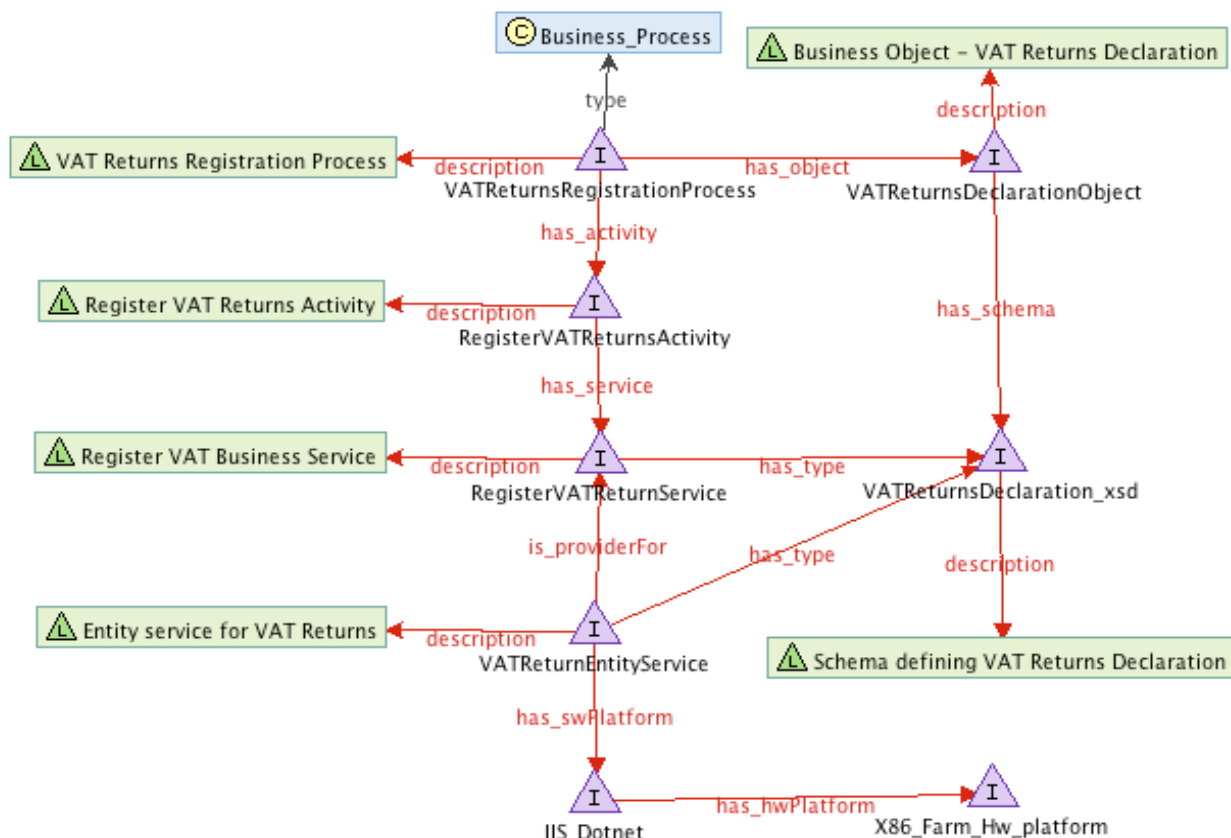


Figure 32: Line-of-Sight - Architecture Domains [Own production]

The business process is once again found at the top of the diagram. Similarly, to the previous figure, the *Register VAT Returns* activity is attached to the process via the *has_activity* property. To the right side, the *VAT Returns Declarations* business object has been related to the process through the *has_object* property. The business service *Register VAT Business Service* is being consumed by the *Register VAT Returns*, through the use of the *has_service* property. The service itself utilises a schema that defines the structure of the *VAT Returns Declarations*. Moreover, the *VAT Returns Declarations* business object is expressed by the *VAT Returns Declaration* schema (via *has_schema*).

At the bottom, it can be seen, that an entity service for VAT returns is a provider for the business service (is_providerFor). The entity service shares schema with the business service. Furthermore, the entity service is hosted on an IIS Dotnet platform, running on an x86 based server-farm. So far, line-of-sight has been demonstrated within the four architectural domains. It is now possible to trace the relations from business process and all the way down to the hardware platform. The model could in theory include as many concepts and attributes as would be desirable. Moreover, it appears evident, that the model reduces complexity by focusing on the understanding of relations between the components in the architecture. Understanding the relations between components, provides a vital first step in being able to perform impact analysis.

Looking in the opposite direction, it is also possible to provide line-of-sight with the strategic domain. Just as changes to components within the four architectural domains has the ability to impact each other, so does the strategic domain interface with the architecture. Whenever strategy elements are being changed, they potentially impact the architecture. Understanding the relationship between the strategy and the four architecture domains provides for improved strategic agility; only by understanding how strategy and architecture components relates to each other, can the architecture be accommodated to meet changed strategic needs. The link between the strategy domains and the architecture domain is seen in figure 33.

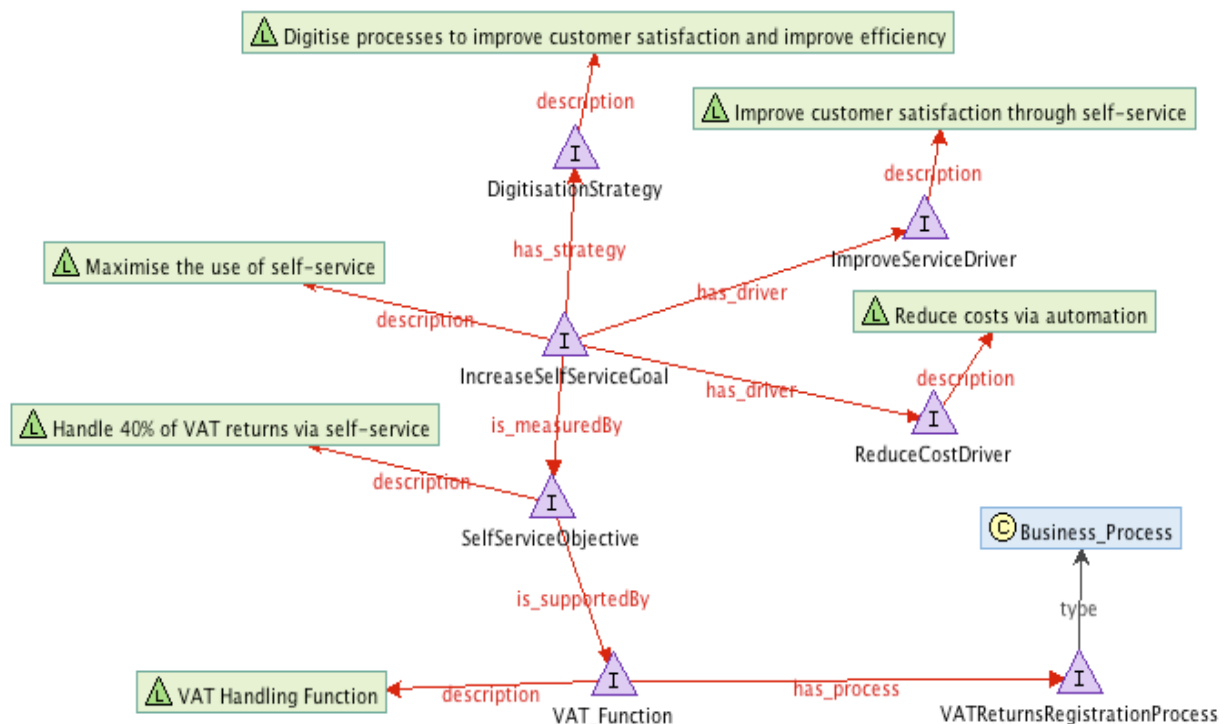


Figure 33: Line-of-sight - Strategy Domain [Own production]

Breaking down the figure from top to bottom, it starts with a digitisation strategy. As the description tells, the strategy is aimed at digitising processes. A goal of the strategy is to maximise the use of self-service. There are two drivers (*has_driver*) behind this goal; to improve customer satisfaction through self-service and to reduce costs via automation. The goal is measured by (*is_measuredBy*) an objective stating that 40% of the VAT returns must be handled via self-service. The self-service objective is to be handled by the VAT function (*is_supportedBy*), which is the owner of the *VAT Returns Registration* process (*has_process*)

To summarise, the line-of-sight use-case has now been demonstrated. The discovery took its beginning with the knowledge of a single node, which in this case was the business process. From this node, it was possible to explore all the related nodes, thus ending up with a topology of the architecture. Moreover, the use of a semantic network as notation offered a huge reduction in complexity.

7.2.3.2 Versioning

The second use-case concerns that of versioning. As it has been argued, a versioning scheme is needed in order to manage and deploy multiple components at the same time. Due to coupling, it is not always possible to unitarily change one component in the architecture at a time, as this would break clients assumptions about a service.

Key competence questions for this use-case, would follow along the line of:

- What is the version history of service x?
- Which version of service x is deployed in Target Architecture version y?

In the example below (figure 34) a new version of the business process has been created. Through the use of the *is_newerVersionOf* property, the two business processes has been linked. In this way, it is possible to trace different versions of the business process back and forth. The new business process has the *changeLog* dataproperty associated, stating that an activity has been added to reject VAT returns. The new activity can further be seen associated to the new process. Finally, the old and the new process shares same version of the three existing activities, which can be seen by both the old and the new business process being related to the activities.⁹⁸

⁹⁸ Whether an enterprise will engage in versioning activities is an entirely different question, but it is not the point here.

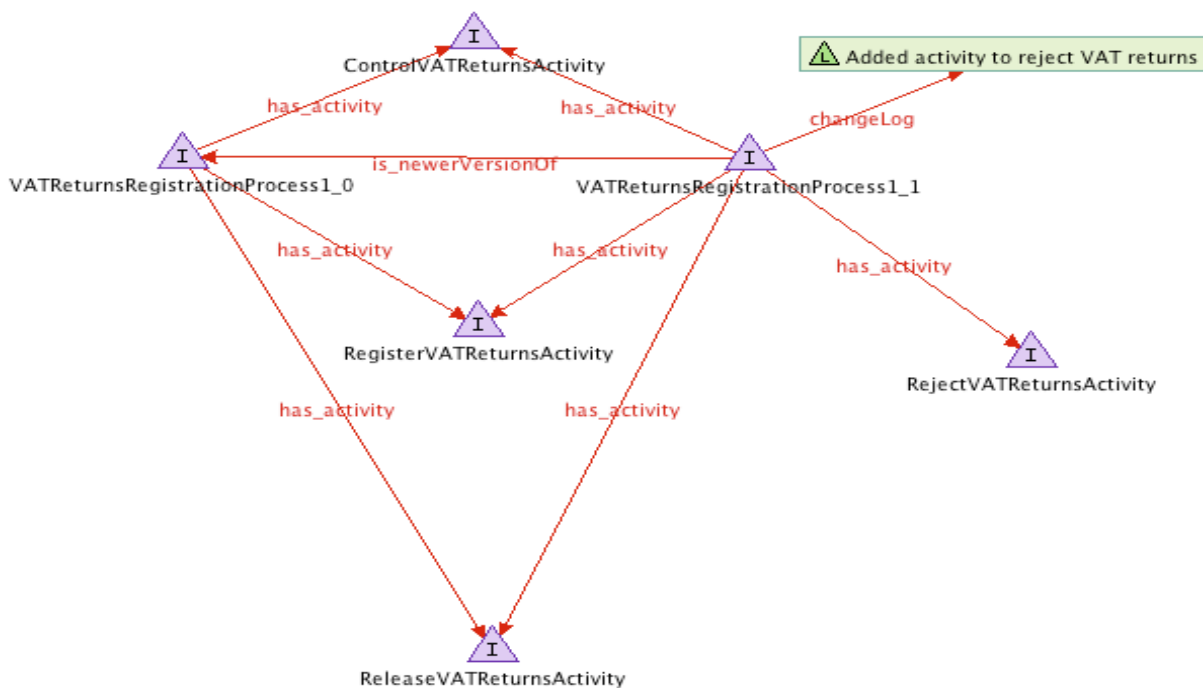


Figure 34: Business Process Versioning [Own production]

Finally, the *is_newerVersionOf* property can be used to denote an updated version of an artefact at any aggregation level. Figure 35 is similar to the previous figure, except that the link between versions is now being shown on the level of the entire architecture, i.e. Baseline Architecture vs. Target Architecture.

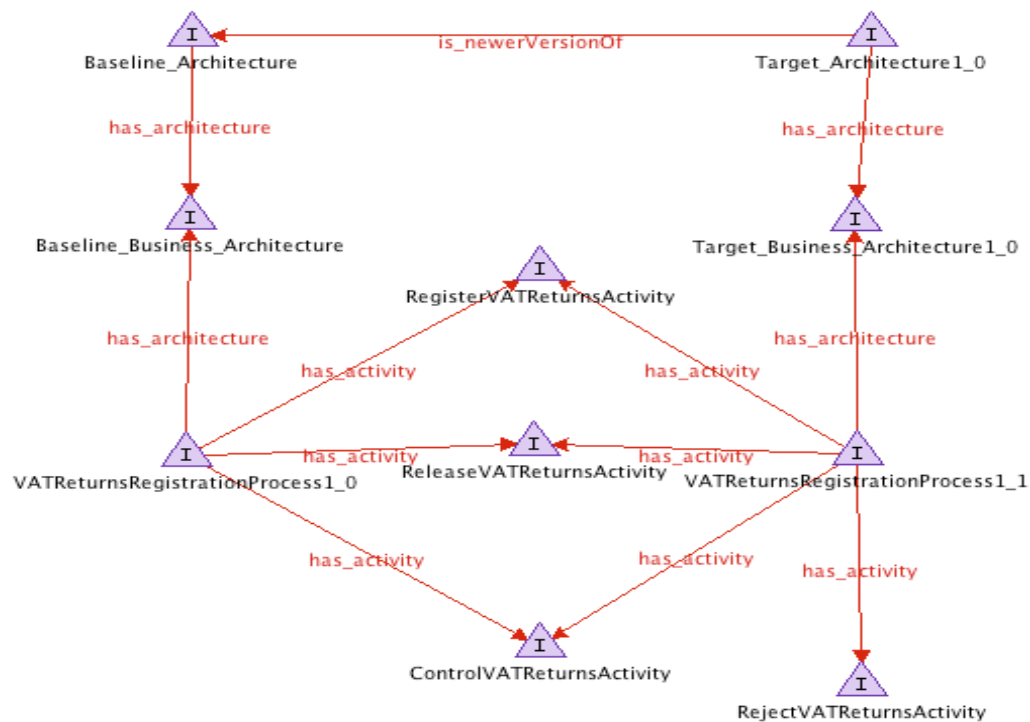


Figure 35: Architecture Versioning [Own production]

In larger architectures versioning solely on individual components and/or on architecture level may be problematic. Versioning solely on the level of individual components is too fine-grained, whereas versioning only on the level of architectures can be considered too coarse-grained.⁹⁹ It is however also possible to define arbitrary aggregate artefacts, such as *models* and *architecture building blocks* (ABB), which are recognised by TOGAF. The different levels of aggregation are not mutually exclusive.

It can thus be concluded, that it is possible to fulfil the simple requirements of the use-case, i.e. to perform versioning on different aggregate levels and to explore their relationships. This means that changes to different components in the architecture can be treated as “blocks” of changes. This does provide some level of isolation, as the architecture can be partitioned into different blocks. This can often be necessary in order to avoid unilateral changes breaking clients assumptions.

⁹⁹ There is however the issue of how to maintain versioning when elements of an aggregate is changed.

7.3 Personal Reflections

In this final part of the chapter, I will detail some of my personal reflections on the case scenario and on the ontology implementation. This small-scale implementation of an ontology does not constitute a full feasibility analysis in itself. But even this, I think there are some important lessons to be learned. So in this section, I am going to be a little more subjective than in the rest of the thesis.

The first conclusion I will draw from this scenario is, that there certainly is a need for a common language to communicate about the components in an architecture. The scenario examples given through the use-case demonstrations were very limited, both in number of concepts defined, and in terms of number of instances defined. Still, even such simple examples do produce a fair amount of meta-data, including the many different object properties. It is apparent, that the amount of meta-data in a BPM-SOA environment, containing many complex business processes and hundreds of services will be staggering. Without having access to integrated meta-data, such architectures will be next to impossible to govern. This highlights the paradox that was described earlier in the thesis; BPM and SOA is supposed to make the architecture more flexible. But as the architecture scales beyond what a small dedicated team of people can comprehend, a much more systematic approach must be taken to enable the understanding of the architecture. Otherwise, architectures that may be flexible from a technological point of view, may be completely sedimented from a managing point of view. There is in my mind no doubt, that initiatives such as BPM and SOA requires a sound grasp of meta-data management.

But if there is a need for a common architecture language is Semantic Web technologies then the right tool? There are several observations that speaks for this. Firstly, as I think it was clearly demonstrated, the data-model of RDFS is extremely versatile. It certainly shows that RDFS was made for knowledge representation, and not for say transactional processing of data. The triple structure is simple, yet powerful enough to express very complex domains. Furthermore, expressing the triples through semantic networks, provides a very efficient way to make generalisations about a domain, thus reducing complexity.

The second observation concerns the way that data and meta-data stored in the same way. This is really one of the strong features of RDFS. The ability to explore the data without having to refer to an externally defined data-model is crucial, as it allows a user to browse the ontology from a point of interest.

Lastly, the combination of integrated data/meta-data and RDFS being an open standard, provides for an extremely high degree of portability. My (albeit limited experience) is that there are few problems in moving the ontology around between different tools.¹⁰⁰ One huge benefit is that the entire ontology (or parts thereof) can be loaded into specialised tools, f. ex. for analysis or visualisation. In this sense, Semantic Web technologies has been a pleasure to work with.

But semantic web technologies are not perfect. It goes without saying, that since RDFS and RDF has been out for some years now without reaching mainstream adoption, then there must be some blind spots. The first major issue in regards to Semantic Web technologies concern the availability of tools. During the course of writing this thesis, I have tried out a large number of different tools and in general found tools to be immature:

- Quite a few of these tools are more proof-of-concepts or beta versions than production ready tools.
- Some tools with similar feature sets have been implemented using different standards, with similar feature sets, but with different syntax's.¹⁰¹ In other cases, only a subset of the capabilities of a standard has been implemented.
- Moreover, the feature set of f. ex. Protégé is in a flux with functionality being removed (such as SPARQL queries) and some plug-ins not loading either in older versions or in newer versions.¹⁰² This is of course a sign of fast-paced development, but perhaps also a sign that ontology development tools has yet to mature.

¹⁰⁰ Apart from the proprietary PAL constructs that I had to remove when converting the ontology to OWL/RDF. See earlier comment.

¹⁰¹ Such as RDF-Gravity relying on RDQL instead of SPARQL

¹⁰² Protégé currently exists in a stable version 3.2, a beta-release of version 3.4 and an alpha version 4.0.

It does also seem that there is a clash of culture and terminology between the Semantic Web people and the enterprise people.¹⁰³ Ontologies and the Semantic Web originates within the world of knowledge representation, philosophy and AI. This certainly shows. Much of the literature and work being done is also still closer to research level, than implementation level. So there is still quite a gap from theory to implementation.

Some movements may be underway to alleviate these issues. A recent report [Provost] on the market for Semantic Web technologies concludes that the market is moving ahead and approaching mainstream. The conclusion of the report states:

“As the Semantic Web industry continues to mature and targeted applications and solutions emerge, the nature of the industry discourse will change. Instead of focusing on extremely low level concepts and terminology used by researchers, a much higher level of discussion will emerge and it will increasingly be driven by business managers.”

[Provost:9]

This change of discourse will be an important element in taking semantic web technologies a step closer to mainstream. The author finally concludes:

“The Semantic Web industry is alive, well, and it's increasingly competitive as a commercial technology. At this point there are too many success stories and too much money being invested to dismiss the technology as non-viable.” [Provost:9]

One can only hope that this is true, and that the potent technologies originating from the Semantic Web initiatives can be put to mainstream use.

¹⁰³ Anyone who has ever followed the SOAP vs. REST debate would not be surprised if such a clash existed.

Module 4 – Conclusion, Perspectives and Criticism

Chapter 8 – Conclusion, Perspectives and Criticism

This chapter marks the end of the thesis. I will start by making the conclusion based on the thesis problem statement. From there, I will set the greater context for the thesis by providing some perspectives. Finally, I will critically self-reflect on the way the research has been done in the thesis.

8.1 Conclusion

The basic tenet of this thesis has been, that enterprises of today are operating in a fast-paced and unpredictable environment. This environment requires enterprises to react much more efficiently to changes. Moreover, I noted that EA, SOA and BPM are being used as strategic initiatives, sometimes deployed as stand-alone initiative and sometimes deployed in combination in order to cope with increasing demands. On basis of these observations, I formulated the research problem as:

How can enterprises integrate EA, BPM and SOA for agility?

I then defined agility as the capability to continuously perform incremental improvements to business processes and underlying systems.

I started out by investigating the relationship between BPM and SOA. Employing BPM and SOA can in many ways be seen as an attempt to close the gap between business and IT. I envisioned a situation in which a business analyst could manage the entire business process life-cycle, with little or no intervention from IT. The business analyst would perform business modelling and then pass on the business process model to the underlying execution platform, which would then be able to execute the SOA services as needed. Such a vision would imply, that it is possible to bridge the gap between business and IT. Any semantic gap would be an impediment to continuous improvements.

Through the model-driven perspective I investigated this reality. During the analysis, I found several semantic gaps. There are still semantic gaps between process modelling and process execution to be found, but these gaps can be bridged with some caveats. Proprietary mappings and extensions are likely to be needed for the foreseeable future. In the future, mainstream unified process analysis and execution languages may solve this problem. Moreover, it can be argued, that there will always be an inherent conflict between modelling for understanding and communication, and modelling for execution.

An even more fundamental semantic gap exists between business modelling and service modelling. An assumption of the BPM-SOA vision is that the business analyst has access to discoverable and re-usable components. This is not always the case. So human cognition, in the form of co-operation between business analysts, solution architects and business developers will still be needed. This is especially relevant in regards to identifying and specifying services, as well as for managing service change. I positioned BPM as a natural tool to be used for identifying and specifying services, however to be augmented with existing asset analysis and identification of additional constraints and non-functional requirements.

As for service change. I became concerned about the possible impact of residual coupling in general, and the impact of semantic coupling in particular. I see the main focus in most SOA work being directed at reducing syntactical coupling. While reducing syntactical coupling is certainly essential to SOA success, it would be costly to ignore the effects of semantic coupling. This would be a major impediment to flexibility. Classic disciplines like impact analysis and change management will not disappear with SOA. On the contrary, achieving agility can only be done with a sound practice of meta-data management and with sound governance procedures.

BPM can be seen as the business case for SOA; by using loosely coupled services as an abstraction layer between business and IT, it is possible to bridge the gap between business and IT in a way that does not embed the business processes within IT-systems. So the true value in combining BPM and SOA does not lie in automated software development, but rather in decomposing business processes and implementation details. The service description is an important abstraction layer between use and implementation. Unfortunately, I see a risk that BPM-SOA will be sold as the next silver-bullet, but will end up being regarded as Case Tools 2.0 due to unrealistic hype.

Caution should be given as to treating BPM and SOA as primarily tactical tools. Rather, to provide sustainable value, adopting BPM and SOA must be based on a strategic fit; the internal capabilities obtained by adopting BPM-SOA should match the strategy of the enterprise. Only by taking the strategic view, can the commitment to such an encompassing paradigm as BPM-SOA be obtained. In particular, I see EA as an important tool for achieving this strategic fit, as it takes an integrated view on strategy, business and technology. During my analysis, I used the TOGAF ADM as an example approach to EA and found the ADM to be a practical methodology for alignment. But this focus on creating alignment also has a downside; EA in general, and the TOGAF ADM in particular does not appear to be very agile methodologies. Rather, the ADM rests on the assumption, that architecture development can be performed on basis of a stable baseline. Changes to the baselines are to be treated as exceptions. Such an approach does not fit well with the need to react as an agile enterprise. I therefore modified the ADM in order to provide a more balance approach. Especially, such an approach requires the architecture governance model being kept separate from the architecture cycle.

This approach does however require full visibility through the architecture artefacts. As the architecture gets decomposed into individual components, each component having its own life-cycle, the ability for any human to comprehend the entire architecture gets lost. Something must provide the bridge from the many domain specific models to a consolidated view of the enterprise. I have argued, that a common language can connect the architecture artefacts, thus providing visibility. Such a common language would help stake-holders understand the relations between the components in the architecture, providing the basis for disciplines such as impact analysis.

The first part of creating such a language would be the establishment of an enterprise meta-model. The enterprise meta-model will be a vocabulary of the important abstract concepts that makes up the enterprise, as well as the types of relations that can be drawn between them. I devised a simple meta-model to illustrate the concept. The second part of defining the language should include devising a way to implement one or more concrete syntax's. Preferably, the concrete syntax's should be implemented in ways that promotes both human understanding and machine-understanding.

I looked at ontologies and ontology representation languages for means to provide a concrete syntax. Ontologies extends taxonomies by also including instances and their relations. In general, I found Semantic Web technologies to have interesting perspectives in relation to the topic of meta-data integration. In particular, I found that the Resource Description Format (RDF) and its schema counterpart (RDFS) had interesting features, making it a suitable lingua franca for meta-data integration. Through the implementation of the simple meta-model, I demonstrated initial feasibility of using Semantic Web technologies for this task.

To finally conclude on this thesis, I see meta-data integration, achieved through the use of a common architecture language, as absolutely vital in integrating BPM, SOA and EA around the theme of agility.

8.2 Perspectives

The purpose of this section is to set the greater context for the thesis, as well as making some forward looking statements.

One of the most interesting areas in this thesis has been the schism between alignment and agility. The relationship between them is not very well understood and during this thesis it has become evident, that the two concepts are very hard to keep separate. An enterprise cannot decide to manage either for agility or for alignment alone; instead a more balanced approach is called for. The schism between agility and alignment is especially interesting in the context of EA. Until now, EA has primarily been a tool for alignment. But I see EA must become more balanced in this regard. This is similar to the idea of Coherency Management, as put forward by [Doucet et al.] The concept of Coherency Management is defined as: *“a logical, orderly and consistent relation of parts to the whole”* [Doucet et. al:2]. The three outcomes of Coherency Management are alignment, agility and assurance. The need to balance agility and alignment is both a threat and an opportunity to EA. To stay relevant in a fast paced world, it is vital that EA becomes better suited at managing for agility also. But this is also a promise; because EA has progressed from information systems architecture to including all domains of the enterprise, EA is now uniquely positioned as a tool for the coherent management of the enterprise. [Doucet et. al:1]

In similar vein, it is quite appropriate to discuss the relationship between the TOGAF ADM and SOA. A fundamental question remains; how fit for SOA is the ADM really? It is quite obvious, that the ADM does not support SOA very well “out-of-the-box”. The ADM can, as seen in chapter 5, be customised to better meet the demands of SOA. But having to make rather sweeping modifications to the ADM, just in order to make it fit for SOA, seems more like a work-around than a solution. As of writing this, we should be pretty close to the release of TOGAF version 9. Little is currently known about what it will look like, except that it should better able to cope with SOA. This is long overdue to say the least.

In particular, I see three things that the next edition of TOGAF should address:

- **Improve business architecture:** The Open Group should work on improving the business architecture. The business architecture was added to TOGAF 8 and in some ways it shows; the business architecture feels “bolted” on top of the three other architecture domains. In particular, the business architecture does not appear to be too well integrated with the other domains.
- **Relax the top-down approach:** The ADM is primarily a top-down approach. Ideally, the ADM should become better at supporting notions such as Service Oriented Analysis and Design (SOAD). In SOAD, the four architecture domains are developed through a mix of top-down and bottom-up approaches, almost as the domains were developed in parallel. This is very much in contrast to the sequential way that the ADM prescribes. A more mixed approach would lead to better opportunities for developing the right requirements in the first place.
- **Establish separate governance:** Finally, in the current incarnation of TOGAF, the governance is too closely linked to the architecture development cycle. Moreover, the governance seems to prioritise the integrity of the ADM cycle. These are elements that needs to be relaxed in the next iteration of TOGAF.

Another general theme in this thesis, came to evolve around the subject of meta-data management and meta-data integration. I fully expect to see a growing awareness in this field the coming years. I demonstrated initial feasibility of using Semantic Web technologies for this task. But it is evident, that much more work is needed in this area, both in terms of tool maturity, and understanding the benefits and challenges of adopting Semantic Web technologies. Performing case studies of early-adopters of Semantic Web technologies would be fruitful.

It is also obvious that The Open Group is attempting to position their MDA framework as an ambitious attempt at defining a coherent framework for managing meta-data. If the Semantic Web approach is lightweight, then the MDA approach can be considered heavy weight. But with ambition also comes complexity and it will be interesting to see if MDA takes off outside the developer community.

If indeed using MDA for EA takes off, then it will open up completely new avenues to explore. One opportunity lies in the Archimate architecture language, which at this time is primarily a graphical notation (concrete syntax) for modelling service oriented enterprise architectures. Archimate is to be aligned with MDA/MOF, which would mean that Archimate could provide both a graphical notation, a serialisation format and a meta-model. This could potentially be a very compelling set-up. The people behind Archimate predicts that in few years time, the MDA will be as important for enterprise architecture as it is for software development. [Lankhorst et al.:29] At this point however, I do not consider MDA prime time for EA. There are also issues with the QVT standard, which is the standard to be used for defining model-to-model relations between MOF-based models. The QVT standard has turned out to be extremely complex and voluminous. In [Stahl & Völter:221] the authors wonders if the QVT standard will survive, and further speculates, that revisions and follow-ups to the standard will appear in the near future, unless the QVT standard is bypassed by the emergence of a de-facto model-to-model standard. [Stahl & Völter:222] The future role of MDA is clearly an area that needs to be followed closely.

In any case, I see the field of meta-data management and integration will start moving towards commoditisation during the next years.

8.3 Criticism

Making a decision as to the research design at the beginning of a thesis, naturally has profound implications on the way the research is being done. In this section, I will reflect critically on my choice of research design. I will perform the critique by addressing these questions:

- Was the research done fairly and objectively?
- Should a case study have been included?
- Was the de-limitation of the thesis too narrow?
- Did the use of TOGAF for EA affect my ability to draw conclusions?

Was the research done fairly and objectively?

In order to produce research that would be regarded as objective as possible, I decided to embed the hermeneutic circle into my research design. This meant, that I would have to continuously shift between looking at the parts and looking at the whole. The concern was that I, as a researcher, already have pre-conceived notions about the subject-matter. This could lead to faulty interpretations, blind spots etc.

Overall, I think I was guided very well by the hermeneutic circle. I was already at the onset of the thesis process aware, that the field I was about to study was characterised by a lack of stringent definitions and interpretations. Moreover, the field is moving forward at a fast pace. I think this awareness has in some ways been a help; I knew I would to contrast different understandings and interpretations of each of the three thesis subject areas, and I knew I would have to understand them as a whole also. In many cases, I have been able to contrast several definitions or interpretations of key concepts to discover its “essence”.

My interpretations of some of the key concepts in this thesis has also changed. during the writing process (especially that of EA and SOA). This at least indicates, that the use of the hermeneutic circle has allowed me to modify some of my previous held assumptions.

But given the nature of the field of study, it would be naïve to believe, that pre-conceived notions, opinions and biases has not at least to some degree affected this thesis. But given the way I have worked with the hermeneutical circle, and given that my interpretation of several key concepts has changed, I think there are reasons to believe, that the bias has had a rather insignificant effect on the research result.

Should a case study have been included?

The second problem concerns my approach to studying the subject; I decided to not include a case study. There are several reasons for this, the primary being that I felt a more theoretical grounded thesis was required. The challenge was to see the three main areas of this thesis as an integrated whole, rather than just as individual parts. In this sense, the purpose was to create a “meta-theory” of the three “sub-theories”. This, I believe, has been a worthwhile goal to strive for.

The risk is however, that through an entirely theoretical study, the thesis becomes too detached from reality, and becomes an example of what mockingly could be called “*academic ivory thinking*”. There are two problems in this regard. The first problem being, that the research produced here has not been validated by practice. The solutions put forth here, will undoubtedly have to be moderated by practice. I have however tried to at least alleviate this problem by performing interviews with domain experts (some of who were also practitioners). These interviews has both been a tremendous inspiration, as well they have helped me moderate some of my views.

The second problem is related to the first; even if the theories put forward here may be “correct”, they may have become too abstract or too idealistic to be implemented in real-life. This is a delicate balance to strive for: on one hand, highly abstract and highly codified knowledge as seen in this thesis, will be easier to diffuse. On the other hand, such knowledge can be too detached from its context. There is no doubt, that the integrated approach to EA, SOA and BPM provided in this thesis, represents a best-case scenario, and an abstract one at that.

But as the purpose of this thesis was to create a meta-theory, I do not see the lack of a case as a deficient. Rather, I see case studies seeking to validate, invalidate or modify the theories put forward in this thesis as the next logical step.

Was the de-limitation of the thesis too narrow?

To ensure focus in this thesis, I decided to focus on agility in relation to business processes and IT-systems. This would leave organisational structure and people out of the equation. This has had some consequences.

The perhaps greatest impact of this decision is seen on the role BPM plays in this thesis. BPM here, is primarily focused on digitisation of business processes (automatisation) or digitisation of the business process management process itself. This is arguably only a subset of what a real-world approach to BPM would include, as an approach to BPM would typically include the human aspects as well. Business processes and IT-systems are however so closely intertwined, that I found it necessary and compelling to treat them together and separately from the two other agility elements.

The selection was necessary to ensure a focus in this thesis, but it is also given that the analysis and suggestions in this thesis only concern a subset of enterprise agility.

Did the use of TOGAF for EA affect my ability to draw valid conclusions?

One of the goals of this thesis was to investigate the role of EA. In this context, I selected the TOGAF framework as the basis for my research. This naturally has consequences.

The problem is perhaps not just about selecting one framework versus another; as it has been shown, there are many different takes on EA, and some of these takes are very different from others. From a hermeneutical angle, it would seem that there are in fact multiple valid interpretations of EA. This is also why I sought to identify some common principles of modern EA in chapter 3.

I believe that the conclusions drawn as to EA in this thesis are valid on a general level, i.e. since there are certain commonalities as to EA, it is also possible to make conclusions as to EA in general. On a more specific level however, some of my observations are of course only applicable to TOGAF and must be seen in that context.

Appendix

List of Appendices

A - Notice on Changed Problem Statement.....	183
B - Indicators of BPM-SOA Interest.....	184
C - Thesis Interviews.....	186
D - The Zachman Framework	187
E - The TOGAF Architecture Development Method	188
F - Note on Model-driven Architecture (MOF & XMI).....	189
G - The Enterprise Meta-model.....	194
H - Meta-model Concepts (Class hierarchy).....	195
I - Meta-model Relations (Object properties).....	200
J - The RDF-Gravity Legend Map.....	201
K - Bibliography.....	202

A - Notice on Changed Problem Statement

The original problem statement contained two problems:

- How can enterprises achieve greater agility by having an integrated and coordinated approach to EA, SOA and BPM?
- How can emerging standards and technologies help enterprises achieve an integrated model of the enterprise?

The problem statements in its final incarnation is:

- How can enterprises integrate EA, BPM and SOA for agility?

The problem statement was changed for a number of reasons. Firstly, because the first part of the original statements [...*greater agility... by having...*] was not formulated very clearly. The second reason was that I found the two thesis questions somewhat disconnected. Lastly, I found a single thesis problem to fit better with the way the analysis was to be performed.

B - Indicators of BPM-SOA Interest

BPM and SOA can and will be pursued as independent initiatives. Some success has been achieved with BPM relying on proprietary middle-ware, whereas large scale SOA success stories seems harder to come by. [Manes]

Customer interest

There are obvious sample and bias problems in using vendor survey as empirical material. The surveys do however show a rather large customer interest in BPM-SOA. Estimates do however vary from survey to survey. According to a BEA report on BPM a “*survey of BEA AquaLogic BPM customers in November 2007 revealed that 68 percent of respondents are connecting BPM and SOA*” [BEA:16] A more modest estimate comes from BPTrends which suggest that 34% of the respondents were utilising BPM and SOA together on projects. [Harmon & Wolf:26]

Vendor consolidation

BPM and SOA vendors are increasingly consolidating in order to provide merged product offerings. Examples of significant deals include:*

- IBM's acquisition of Holosofx
- Oracle's acquisition of Collaxa
- Tibco's purchase of Staffware,
- BEA's acquisition of Plumtree and Fuego
- Tibco's purchase of Staffware
- IBM's acquisition of Telelogic
- Oracle's acquisition of BEA

* *In parts based on [Kamoun:1]*

Awareness

The final indicator of interest is the increasing level of awareness around BPM and SOA. Examples are Gartner's prediction that SOA and BPM are converging and that BPM will be the driver for SOA implementations. [Seeley, 2006] Prolific bloggers on process management and service-oriented architecture such as Ismael Ghalimi hail BPM as "the killer application for SOA" [Ghalimi]

The body of scholarly research and academic literature on this topic is however scarce. Much of the knowledge about BPM-SOA is rooted among practitioners (adopters, consultancies or vendors) among . Furthermore, the existing literature often takes either the BPM or the SOA approach, while just touching on the opposite angle of research.

C - Thesis Interviews

<u>Name</u>	<u>Employer</u>	<u>Title</u>	<u>Date</u>
Rasmus Knippel	Danske Bank	Enterprise Architect	08/04-2008
Anders Mortensen	Bizcon	CEO & CSO	05/05-2008
Henrik Søgaard	Bizcon	COO & CIO	05/05-2008
Kuno Brodersen	Qualiware	CEO	18/03-2008
Frank Carvalho	Skat	Chief Architect	28/04-2008

D - The Zachman Framework

ENTERPRISE ARCHITECTURE: A FRAMEWORK™

ZIFA
Zachman Institute
for Framework Advancement

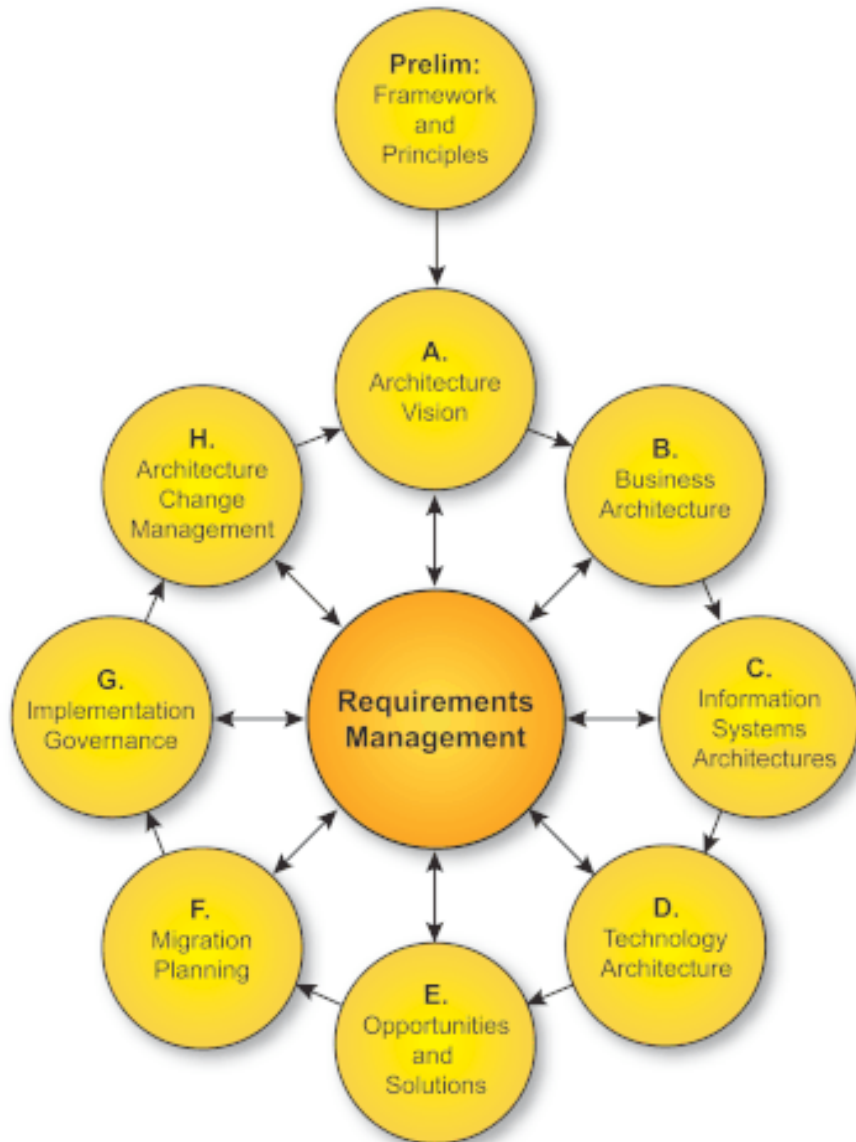
PHONE (810) 231-0531
FAX: (810) 231-6631
www.zifa.com
10895 Lakepointe Drive
Pineknay, MI 48169

WHAT		HOW	WHERE	WHO	TIME	WHY
DATA	FUNCTION	NETWORK	PEOPLE	TIME	MOTIVATION	
SCOPE (contextual) Planner	List of Things Important to the Business Entity = Class of Business Thing	List of Processes the Business Performs Process = Class of Business Process	List of Locations in Which the Business Operates Node = Major Business Location	List of Organizations Important to the Business People = Major Organizational Unit	List of Events/Cycles Significant to the Business Time = Major Business Event/Cycle	SCOPE (contextual) Planner Ends/Means = Major Business Goal/Strategy
BUSINESS MODEL (conceptual) Owner	e.g., Semantic Model Entity = Business Entity Relationship = Business Relationship	e.g., Business Process Model Process = Business Process IO = Business Resource	e.g., Business Logistics System Node = Business Location Link = Business Linkage	e.g., Work Flow Model People = Organization Unit Work = Work Product	e.g., Master Schedule Time = Business Event Cycle = Business Cycle	BUSINESS MODEL (conceptual) Owner e.g., Business Plan End = Business Objective Means = Business Strategy
SYSTEM MODEL (logical) Designer	e.g., Logical Data Model Entity = Data Entity Relationship = Data Relationship	e.g., Application Architecture Process = Application Function IO = User Views	e.g., Distributed System Architecture Node = I/S Function (Processor, Storage, etc.) Link = Line Characteristics	e.g., Human Interface Architecture People = Role Work = Deliverable	e.g., Processing Structure Time = System Event Cycle = Processing Cycle	SYSTEM MODEL (logical) Designer e.g., Business Rule Model End = Structural Assertion Means = Action Assertion
TECHNOLOGY MODEL (physical) Builder	e.g., Physical Data Model Entity = Segment/Table/etc. Relationship = Pointer/Key/etc.	e.g., System Design Process = Computer Function IO = Data Element/Set	e.g., Technology Architecture Node = Hardware/Software Link = Line Specifications	e.g., Presentation Architecture People = User Work = Screen Formats	e.g., Control Structure Time = Execute Cycle = Component Cycle	TECHNOLOGY MODEL (physical) Builder e.g., Rule Design End = Condition Means = Action
DETAILED REPRESENTATIONS (out-of-context) Subcontractor	e.g., Data Definition Entity = Field Relationship = Address	e.g., Program Process = Language Statement IO = Control Block	e.g., Network Architecture Node = Address Link = Protocol	e.g., Security Architecture People = Identity Work = Job	e.g., Timing Definition Time = Interval Cycle = Machine Cycle	DETAILED REPRESENTATIONS (out-of-context) Subcontractor e.g., Rule Specification End = Sub-equation Means = Step
FUNCTIONING ENTERPRISE		e.g.: DATA	e.g.: FUNCTION	e.g.: NETWORK	e.g.: ORGANIZATION	e.g.: STRATEGY
FUNCTIONING ENTERPRISE		FUNCTIONING ENTERPRISE				

© John A. Zachman

THE ZACHMAN FRAMEWORK FOR ENTERPRISE ARCHITECTURE

E - The TOGAF Architecture Development Method



F - Note on Model-driven Architecture (MOF & XMI)

Standards for modelling are necessary to ensure not only the interoperability across tools, applications, and repositories, but also across modelling artefacts. Until now, modelling practices have been characterised by a low degree of interoperability of modelling languages. MDA in general, and the Meta-object Facility (MOF) is set to change that.

MOF is an OMG standard related to MDA, which claims to be “*an extensible model driven integration framework for defining, manipulating and integrating metadata and data in a platform independent manner.*” [OMG, 2008] The MOF specification defines an abstract language, and a framework for specifying, constructing, and managing technology neutral meta-models. Meta-models models are used to model the modelling languages themselves [MOF:29] So the MOF language is a minimal set of constructs, that can be used to model other modelling languages. [Gasevic, Djuric & Devedzic:114] The MOF language is essentially a language, that can be used to define the abstract syntax (or meta-models) of modelling languages. Examples of modelling languages that has meta-models that conforms to MOF, are the UML, the Common Warehouse Model (CWM), and the MOF language itself¹⁰⁴. The MOF language is a subset of UML, and in MOF 2.0 the modelling concepts have been unified with UML2.¹⁰⁵ [MOF:7] This also means that UML based models are in themselves MOF compliant.

By establishing MOF as a single language for the specification of meta-models, MOF becomes a bridge that ensures a modest degree of commonalities between these languages. This provides the opportunity to map model elements in a model, to model elements in other model types.¹⁰⁶ This even if the languages involved are very different in purpose. This is a critically important feature in achieving the ability to perform transformations. MOF seems to pave a critical path in creating a modelling value-chain.

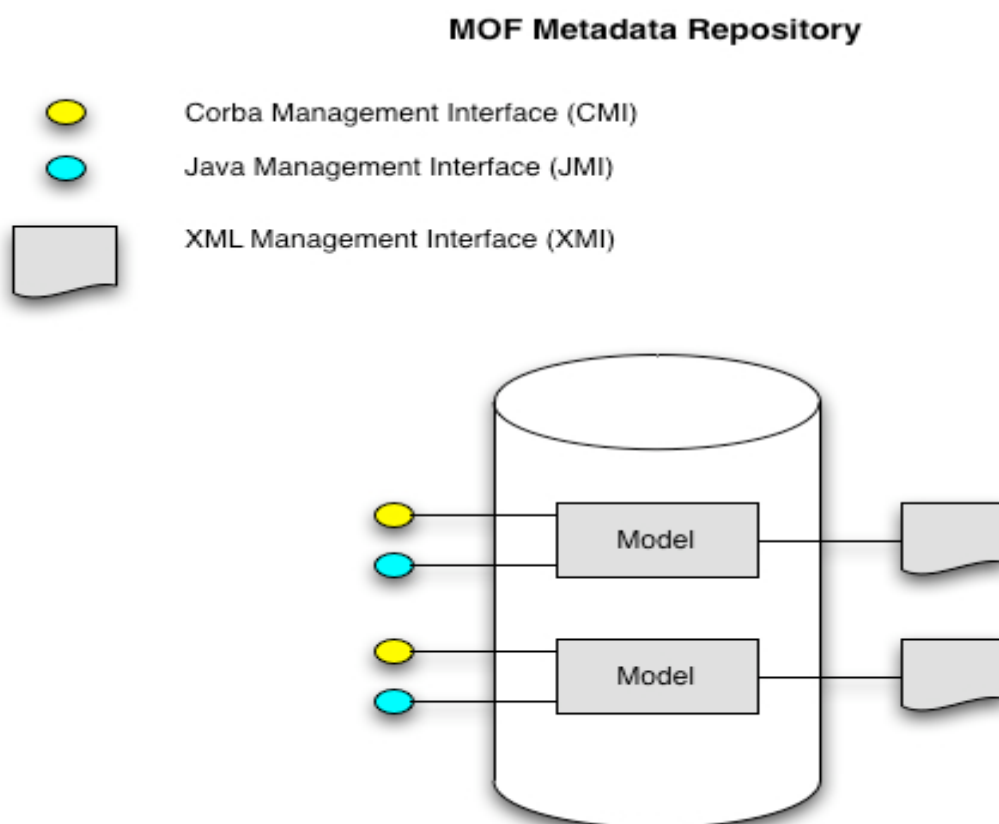
¹⁰⁴ The relationship is recursive as MOF is self-described

¹⁰⁵ The MOF 2.0 standard is adopted, but pending pre-finalisation

¹⁰⁶ Assuming the mappings are performed on meta-model level, rather than model-instance level.

MOF based meta-models, do however only specify the abstract syntax of a model language. To create a usable DSML, it is necessary to devise some way of defining the concrete syntax. UML Profiles provides such opportunity; UML Profiles work by supporting the adaption or extension of UML to fit professional or technical domains. UML Profiles consists of three different types of artefacts; stereotypes, tagged values and constraints. [Stahl & Völter:93] An example of an UML profile is the “*UML Profile and Metamodel for Services*” (UPMS), which seeks to define both the abstract, and the concrete syntax of a modelling language for services in SOA. [UPMS] A further advantage of UML profiles is, that UML profiles can be loaded in UML tools as plug-ins.

The MOF standard is not only interesting as a meta-language for defining modelling languages. By defining a set of standard interfaces (APIs), the MOF also becomes a framework for implementing repositories that can be used to hold, and manage models. models. Figure zz illustrates the MOF repository.



The three API interfaces are:

- **CORBA Meta-data Interface (CMI):** Defines how to manage MOF models as CORBA objects
- **Java Meta-data Interface (JMI):** Defines how to manage MOF models as Java objects. Provided by SUN systems.
- **XML Meta-data Interchange (XMI):** XMI provides mapping from MOF to XML. This makes MOF interoperable, by allowing a standardised way to persist MOF-based models to XML.¹⁰⁷

By providing mappings to transform MOF meta-models into APIs (standard interfaces), the APIs allows for interoperable repositories, despite differences in underlying implementation technologies. The three APIs do however not result in the same set of opportunities. The CMI and the JMI interfaces are APIs in the classical sense, as they provide access to certain functionality at the back-end. Despite the well-thought intentions behind the MOF APIs, there are some issues with the way they have been designed. As [Blanc, Bouzitouna & Gervais] points out, the APIs are too low level, and the APIs generally needs to be extended, in order to provide the appropriate functionality. The MOF APIs does f. ex. not concern non-functional issues, such as persistence, security, and performance. [Blanc, Bouzitouna & Gervais: 120] It is clear, that the repository interfaces only provides a starting point, and that more work needs to be done, f. ex. by defining higher-level APIs.

Of the three API's, XMI seems by far the most interesting. XMI in contrast, could be considered a serialisation format for MOF based models. The XMI standard describes the XMI document structure, called the XMI model. The XMI model is essentially an instance of MOF, described in a way that allows MOF meta-models¹⁰⁸ to be serialised into the XMI document format. [XMI:8] This is an important feature, as it allows MOF to leverage the dominant position of XML.

¹⁰⁷ Since the release of version 2, XMI has included support for serialising diagram layout which is essential for tool-to-tool transfer [Stahl & Völter:292]

¹⁰⁸ And MOF-based models

Another interesting feature is that XMI allows for the creation of XML Schemas for the meta-models. XMI also holds the distinct feature, that is defines the permissible structures for each language (meta-model) An XML Schema would only be able to constrain syntax, but cannot define all possible structures and their relationships. [Borenstein & Fox] As such, the XMI becomes a standard for exchanging MOF based meta-models and models in XML. This allows XMI to leverage the ubiquitousness of XML, rather than to rely on Java or Corba interfaces. Any tool being able to read and write XML, would in theory be capable of working with XMI files.

Unfortunately, the ability to use XMI as a serialisation format has some problems. [Lundell et al.] has made an empirical study as to the interchange of XMI models in an heterogenous tool environment. The study consisted of two tests. The first test was made to investigate the issues around working with different version of XMI. As of writing this, there are several versions of XMI recognised by OMG; 1.0, 1.1, 1.2, 2.0, 2.01, and 2.1. [Lundell et al.:621] A simple UML Class diagram was tested for interoperability between five tools that supported XM version 2.0 or later, and another set of legacy tools supporting only earlier versions of XMI. The result can be seen in figure 36 The non-coloured cells are combinations where tools supports same versions of XMI. Grey cells are combinations where round-tripping is not expected to work, since XMI versions differ.

Import → Export ↓	Borland	Eclipse	Rational	MagicDraw	UModel
ArgoUML	<i>Failed</i>	<i>Failed</i>	<i>Failed</i>	Failed	Failed
Fujaba	Successful	<i>Failed</i>	<i>Failed</i>	Failed	Failed
Umbrello	Failed	<i>Failed</i>	<i>Failed</i>	Failed	Failed
Artisan	Failed	<i>Failed</i>	<i>Failed</i>	Failed	Failed
Poseidon	Failed	<i>Failed</i>	<i>Failed</i>	Successful	Failed
Rhapsody	<i>Failed</i>	<i>Failed</i>	<i>Failed</i>	Failed	Failed
Rose 1.0	Failed	<i>Failed</i>	<i>Failed</i>	Failed	Failed
Rose 1.1	Failed	<i>Failed</i>	<i>Failed</i>	Failed	Failed
Visio	<i>Failed</i>	<i>Failed</i>	<i>Failed</i>	Failed	Failed

Figure 36: UML Model Exchange [Lundell et al.:626]

The second test concerns one-way exchange between the tools that supports XMI version 2 and above. Figure 37 depicts the result of the second test. In general, results were found to be much better, as only MagicDraw, and UModel exhibited import problems. Furthermore, a round-tripping test done with the tools supporting one-way exchange did show, that tools supporting one-way exchange, did also successfully interexchange models two-way. An important caveat however was, that no success was a (for the tools that did support XMI 2.1)¹⁰⁹ and XMI 2.0. This indicates, that backwards compatibility has been lost, between XMI 2.0 and 2.1, at least at test time.

Import → Export ↓	Borland	Eclipse	Rational	MagicDraw	UModel
Borland	Successful	Successful	Successful	Failed	Failed
Eclipse	Successful	Successful	Successful	Failed	Failed
Rational	Successful	Successful	Successful	Failed	Failed
Magic- Draw	Failed	Failed	Failed	Successful	Successful
UModel	Failed	Failed	Failed	Successful	Successful

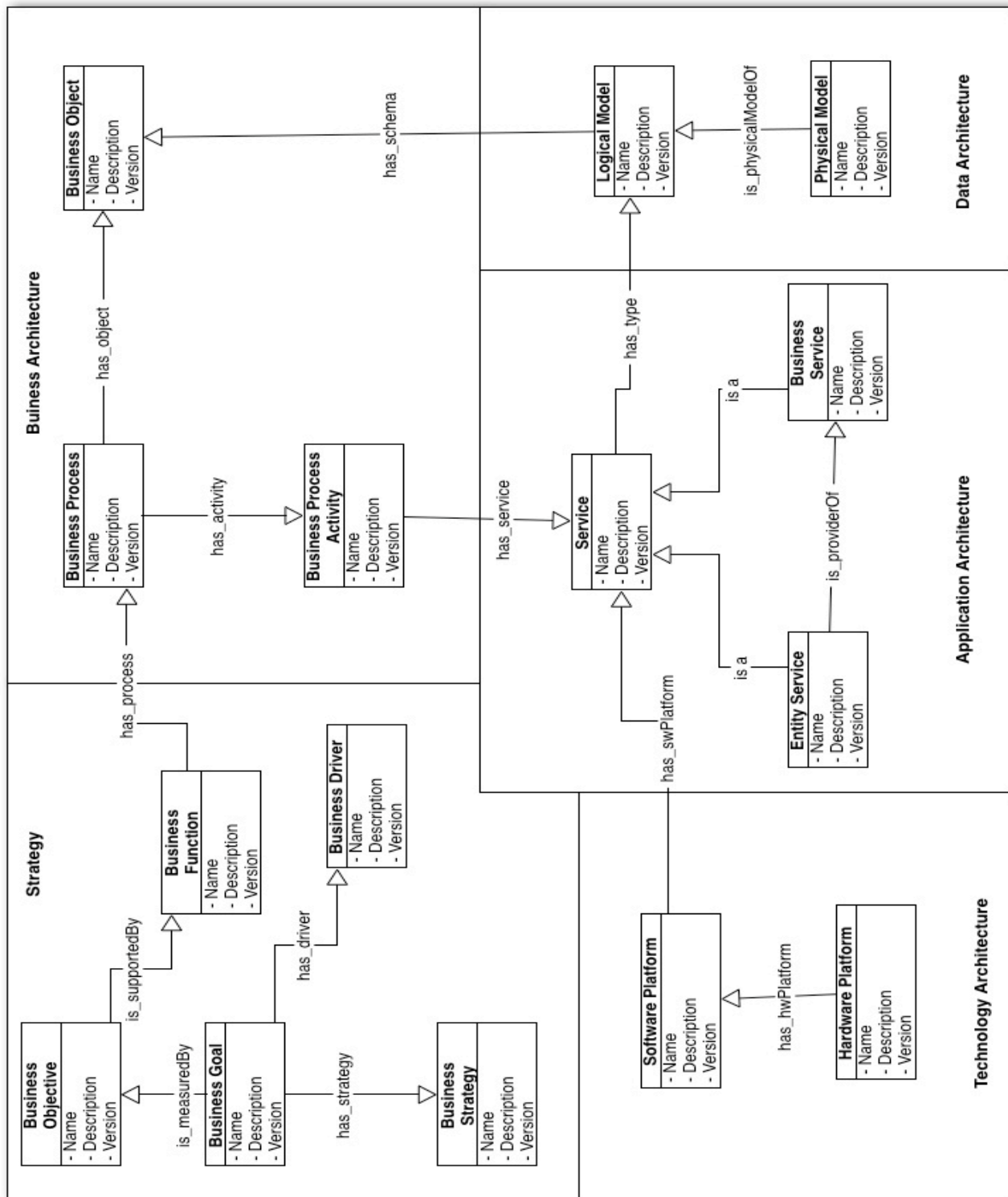
Figure 37: UML Model Exchange XMI 2.0+ [Lundell et al.:628]

It should also be noted, that these tests were only attempts at exchanging simple UML class diagrams. No attempts at exchanging DSML's was attempted, and one can only speculate the result of such test would. So as can be concluded from these test, compatibility between tools and XMI versions leaves much to desired. The matter of integrating information via XMI is further complicated by the fact, that much information will not be stored MOF based languages.¹¹⁰ A heterogeneous information environment is likely to be the default. One such example is the BPMN standard, which of writing this, still does not have standardised serialisation format (let alone a MOF meta-model).

¹⁰⁹ For the tools that supported XMI 2.1

¹¹⁰ And thus be persisted to XMI files or a MOF based repository.

G - The Enterprise Meta-model



H - Meta-model Concepts (Class hierarchy)

* Indicates that the concept has been added to the Ontology

** The term is used in a different way in TOGAF

Strategy

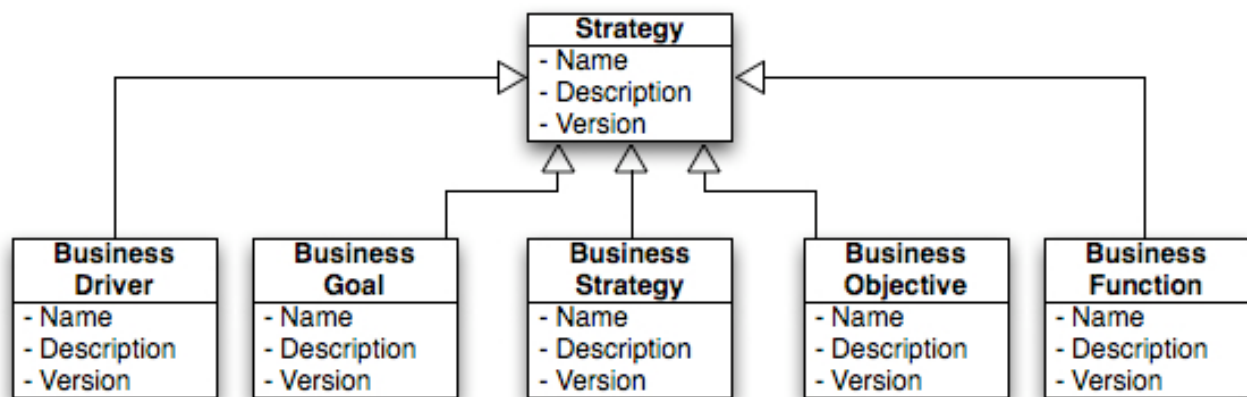


Figure 38: Strategy Class Hierarchy [Own production]

<u>Concept/Class</u>	<u>Description</u>
Business Driver	Drives business goals
Business Goal	What the enterprise plans to accomplish
Business Strategy	How the enterprise plans to achieve goals and objectives
Business Objective	What the enterprise plans to accomplish (usually identified by metrics)
Business Function	A decomposition of the major functional areas of the enterprise into functions.

Business Architecture

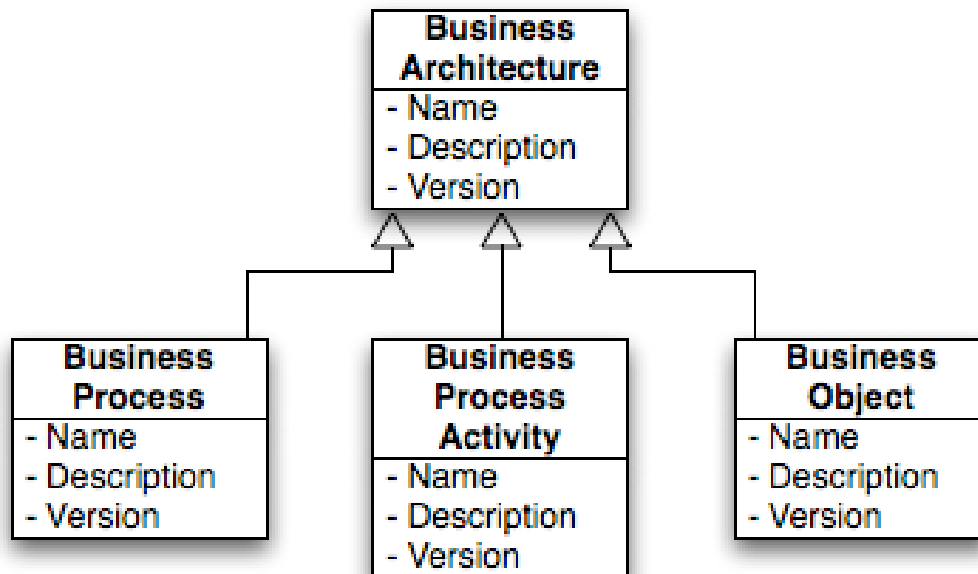


Figure 39: Business Architecture Class Hierarchy [Own production]

Concept/Class	Description
Business Process *	A set of one or more linked procedures or activities which collectively realise a business objective or policy goal
Business Process Activity *	Collections of tasks, that contributes or adds value to process goals.
Business Object *	Abstract or conceptual models of real world entities in the business domain

Application Architecture

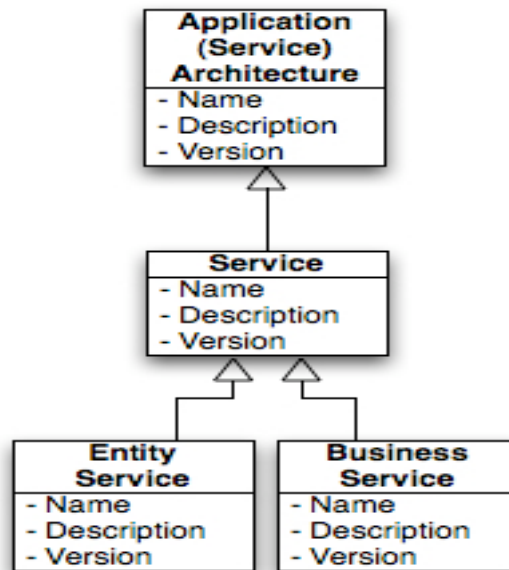


Figure 40: Application Architecture Class Hierarchy

Concept/Class	Description
Service*	Collection of capabilities
Business Service**	Collections of capabilities used to support business processes
Entity Service*	Collections of capabilities used to manage data

Data Architecture

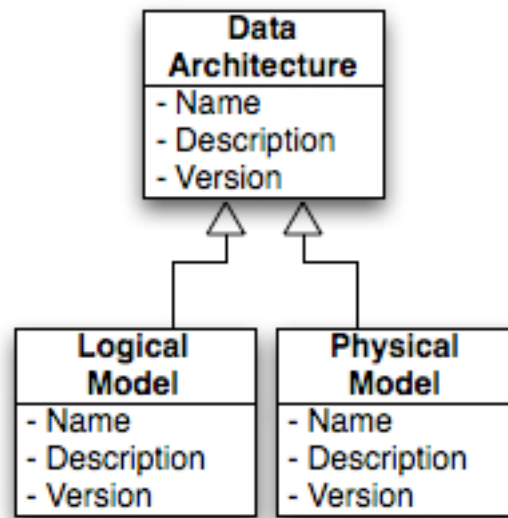
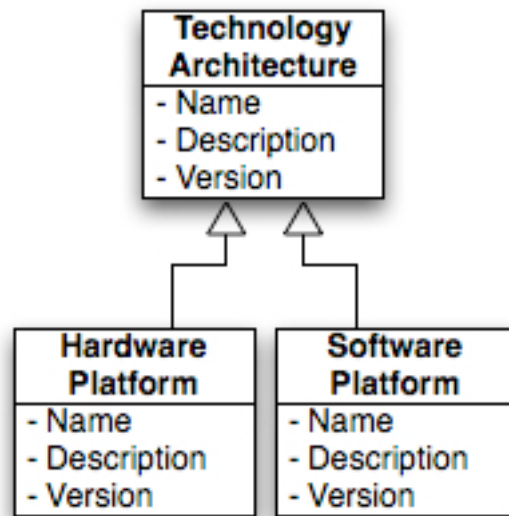


Figure 41: Data Architecture Class Hierarchy [Own production]

<u>Concept/Class</u>	<u>Description</u>
Logical Data Model	Logical views of the actual data of interest from the application point of view. Expressed by the use of a schema.
Physical Data Model	Physical views of the actual data of interest from the application point of view (f. ex. physical database schema)

Technical Architecture



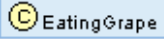
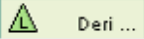


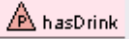

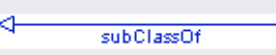
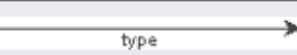

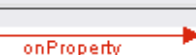
<u>Concept/Class</u>	<u>Description</u>
Hardware platform	Physical hosting platform
Software platform	Software hosting platform

I - Meta-model Relations (Object properties)

<u>Name</u>	<u>Description</u>	<u>Domain</u>	<u>Range</u>
is_measuredBy	Objectives are goals quantified by a metric	Business Objective	Business Goal
is_physicalModelOf	The physical model implements the logical model	Logical Model	Physical Model
is_providerOf	Entity services provides data services to business services	Business Service	Entity Service
is_supportedBy	Business functions supports the realisation of business objectives	Business Function	Business Objective
has_driver	Indicates drivers that contributes to a goal	Business Driver	Business Goal
has_hwPlatform	Physical platform hosting a software platform	Software Platform	Hardware Platform
has_object	Business object is related to a business process	Business Object	Business Process
has_process	Indicates that a business function is being served by a business process	Business Process	Business Function
has_schema	A business object can have logical representations in form of a schema (f. ex. xsd)	Business Object	Logical Model
has_service	A business process activity consumes a business service	Service	Business Process Activity
has_strategy	Business goals are linked to strategy / strategic initiatives	Business Strategy	Business Goal
has_swPlatform	Represents a software platform hosting services	Software Platform	Hardware Platform
has_type	Service relies on schemas for the definition of input/output types	Logical Model	Service

J - The RDF-Gravity Legend Map

Reproduced from: http://semweb.salzburgresearch.at/apps/rdf-gravity/user_doc.html

	Refers to concepts or Classes. Here "EatingGrape" is a concept.
	Refers to a literal value (string, integer) etc.
	Refers to Anonymous nodes
	This refers to instances. Any anonymous node which has rdf:type property associated to a Class is also shown as an instance.
	Refers to Properties
	Refers to URI strings which cannot be identified as any of the above items
	Blue edges refer to rdfs:subClassOf property.
	Black edges refer to the rdf:type property.
	Dotted Edges refer to the rdf:comment property.
	Rest all other properties are marked as red in color.

K - Bibliography

Books

[Bernard]: **"An Introduction To Enterprise Architecture"** / Scott A. Bernard / 2005 / 2nd Edition / AuthorHouse

[Bloomberg & Schmelzer]: **"Service Orient or Be Doomed!: How Service Orientation Will Change Your Business"** / Jason Bloomberg & Ronald Schmelzer / 2006 / Wiley

[Denzin & Lincoln]: **"The Sage Handbook of Qualitative Research"** / N.K. Denzin & Y. S. Lincoln / 2005 / Sage Publications / 3rd Edition

[Dietz]: **"Enterprise Ontology"** / Jan L. G. Dietz / 2006 / Springer

[EABOK], **"Guide to the (Evolving) Enterprise Architecture Body of Knowledge"** / Mitre Corporation / 2004 / Mitre Corporation

[Gasevic, Djuric & Devedzic]: **"Model Driven Architecture and Ontology Development"** / Dragan Gasevic, Dragan Juric & Vladan Devedzic / 2006 / Springer-Verlag

[Erl, 2005]: **"Service-Oriented Architecture (SOA): Concepts, Technology, and Design"** / Thomas Erl / 2005 / Prentice Hall

[Erl, 2008]: **"SOA Principles of Service Design"** / Thomas Erl / 2008 / Prentice Hall

[Friedman]: **"The World Is Flat: A Brief History of the Twenty-first Century"** / Thomas L. Friedman / 2005 / Farrar, Straus and Giroux

[Harrison-Broninski]: **"Human Interactions: The Heart And Soul Of Business Process Management: How People Really Work And How They Can Be Helped To Work Better"** / 2005 / Keith Harrison-Broninski / Meghan-Kiffer Press

[Jeston & Nelis, 2008, 1], "**Business Process Management – Practical Guidelines to Successful Implementations**" / John Jeston & Johan Nelis / 2008 / 2nd Edition / Elsevier Ltd.

[Juric & Pant]: "**Business Process Driven SOA using BPMN and BPEL**" / Matjaz B. Juric & Kapil Pant / 2008 / Packt Publishing

[Kaye]: "**Loosely Coupled**" / Doug Kaye / 2003 / RDS Press

[Khoshafian, 2007]: "**Service Oriented Enterprises**" / 2007 / Setrag Khoshafian / Auerbach Publications

[Lankhorst et al.]: "**Enterprise Architecture at Work: Modelling, Communication and Analysis**" / Marc Lankhorst et al. / 2005 / Springer

[Larman]: "**Applying UML and Patterns**" / Craig Larman / 2004 / Prentice Hall / 3rd Edition

[Passin]: "**Explorers Guide to the Semantic Web**" / Thomas B. Passin / 2004 / Manning Publications

[Rossberg & Redler]: "**Pro Scalable .NET 2.0 Application Designs**" / Joachim Rossberg & Rickard Redler / 2006 / Apress

[Schekkerman]: "**How to survive in the jungle of Enterprise Architecture Frameworks**" / Jaap Schekkerman, 2004 / 2nd Edition / Trafford

[Smith & Fingar]: "**Business Process Management – The Third Wave**" / Howard Smith & Peter Fingar / 2006 / Meghan Kiffer Press

[Swebok]: "**Guide to the Software Engineering Body of Knowledge**" / IEEE / 2004 / IEEE

[Stahl & Völter]: "**Model-Driven Software Development**" / Thomas Stahl & Markus Völter / 2006 / John Wiley & Sons

[Weill, Ross & Robertson]: "**Enterprise Architecture as Strategy**" / Jeanne W. Ross, Peter Weill & David C. Robertson / 2006 / Harvard Business School Press

Websites and Blogposts

[Arsanjani]: **"Service-oriented modeling and architecture"** / Ali Arsanjani / <http://www.ibm.com/developerworks/library/ws-soa-design1> / Last accessed 22/07-2008

[Baurmann]: **"Why OWL triples matter: The Portable Ontology Revolution in Domain Knowledge Representation"** / Stu Baurmann / 2005 / <http://www.xmlexpertise.com/392.html> / Last access 22/09-2008

[Borenstein & Fox]: **"XMI and the Many Metamodels of Enterprise Metadata"** / Joram Borenstein & Joshua Fox / <http://www.idealliance.org/proceedings/xml05/ship/39/XMIAndMetamodels.HTML> / Last access 22/07-2008

[BPMN, 2005]: **"Business Process Modeling Notation (BPMN) Information"** / BPMN.org / <http://www.bpmn.org/Documents/FAQ.htm#Technical> / Last access 22/10-2008

[EMF]: **"Eclipse Modeling framework (EMF)"** / <http://www.eclipse.org/modeling/emf> / Last access 22/07-2008

[ESW]: **"LargeTripleStores"** / <http://esw.w3.org/topic/LargeTripleStores> / Last access 21/10-2008

[FEA-RMO]: **"Federal Enterprise Architecture Reference Model Ontology (FEA-RMO)"** / <http://web-services.gov/fea-rmo.html> / Last access 22/07-2008

[Fowler, 2006]: **"Semantic Diffusion"** / Martin Fowler / <http://martinfowler.com/bliki/SemanticDiffusion.html> / Last access 22/07-2008

[Gartner, 2007]: **"Gartner EXP Survey of More than 1,400 CIOs Shows CIOs Must Create Leverage to Remain Relevant to the Business"** / Gartner / <http://www.gartner.com/it/page.jsp?id=501189> / Last access 22/07-2008

[Gartner, 2008]: **"Gartner EXP Worldwide Survey of 1,500 CIOs Shows 85 Percent of CIOs Expect "Significant Change" Over Next Three Years"** / <http://www.gartner.com/it/page.jsp?id=587309> / Last access 22/07-2008

[Ghalimi]: "**BPM is SOA's Killer Application**" / Ismael Ghalimi / <http://itredux.com/2006/08/13/bpm-is-soas-killer-application> / Last access 23/07-2008

[Guttman]: "**Linking BPM and SOA - It Shouldn't Just Be Magic**" / Michael Guttman / <http://mda-soa.blogspot.com/2006/06/linking-bpm-and-soa-it-shouldnt-just.html> / Last access 22/10-2008

[Guttman & Matthews]: "**Forrester Research Claims MDA is DOA!**" / Michael Guttman and Jason Matthews / <http://mda-soa.blogspot.com/2006/03/forrester-research-claims-mda-is-doa.html> / Last access 22/07-2008

[Hendler]: "**Agents and the Semantic Web**" / James Hendler / <http://www.cs.umd.edu/~hendler/AgentWeb.html> / Last access 22/07-2008

[ISO42010]: "**Recommended Practice for Architectural Description of Software-Intensive Systems**" / <http://www.iso-architecture.org/ieee-1471/index.html> / Last access 22/07-2008

[Jones]: "**Why BPM Screws up SOA**" / Steve Jones / <http://service-architecture.blogspot.com/2007/06/why-bpm-screws-up-soa.html> / Last access 28/08-2008

[Lhotka]: "**Semantic coupling: the elephant in the SOA room**" / Rockford Lhotka / <http://www.lhotka.net/weblog/SemanticCouplingTheElephantInTheSOARoom.aspx> / Last access 22/09-2008

[Malik]: "**SOA and the CISR Operating Models**" / Nick Malik / <http://blogs.msdn.com/nickmalik/archive/2007/10/12/soa-and-the-cisr-operating-models.aspx> / Last access 22/09-2008

[Manes]: "**Looking for SOA success stories**" / Anne Thomas Manes / <http://apsblog.burtongroup.com/2008/03/looking-for-soa.html>
/ Last access 19/04-2008

[Manifesto]: "**Principles behind the Agile Manifesto**" / <http://agilemanifesto.org> / Last access 27/07-2007

[McComb]: "**The Enterprise Ontology**" / James McComb / <http://www.tdan.com/view-articles/5016> / Last access 22/07-2008

[OMG, 2008]: "**Catalog of OMG Modeling and Metadata Specifications**" / Object Management Group /

http://www.omg.org/technology/documents/modeling_spec_catalog.htm / Last acces
2308/2008

[OpenGroup]: "**TOGAF 8 Ontology draft**" / The Open Group /

<http://www.opengroup.org/projects/soa-ontology/doc.tpl?CALLER=index.tpl&gdid=11367> /

Last access 17/09-2008

[Rugg]: "**SOA: The New Tower of Babel**" / Ken Rugg / 2008

http://blogs.progress.com/soa_infrastructure/2008/07/soa-the-new-tow.html / Last access 10/08-2008

[SAP]: "**Process Management Lifecycle (PML)**" / SAP /

<https://www.sdn.sap.com/irj/sdn/bpx-cycle> / Last access 21/09-2008

[Seeley, 2006]: "**SOA and BPM headed for convergence, says Gartner**" / Rich Seeley

"http://searchsoa.techtarget.com/news/article/0,289142,sid26_gci1235127,00.html" / Last access 19/04-2008

[Seeley, 2008]: "**SOA lifecycle: What are we talking about?**" / Rich Seeley /

http://searchsoa.techtarget.com/news/article/0,289142,sid26_gci1213362,00.html / Last access 28/08-2008

[Silver, 2006]: "**Thoughts on BPMN 2.0**" / Bruce Silver /

<http://www.brsilver.com/wordpress/2006/07/06/thoughts-on-bpmn-20> / Last access 22-07-2008

[Silver, 2008]: "**Which Way for BPMN?**" / Bruce Silver /

<http://www.bpminstitute.org/articles/article/article/bpms-watch-which-way-for-bpmn.html> /

Last access 22/10-2008

[Sowa]: "**Semantic Networks**" / John F. Sowa / <http://www.jfsowa.com/pubs/semnet.htm> /

Last access 22/07-2008

[Temnenco]: "**TOGAF or not TOGAF: Extending Enterprise Architecture beyond RUP**" / Vitalie Temnenco /

<http://www.ibm.com/developerworks/rational/library/jan07/temnenco/index.html> /

Last access 22/07-2008

[W3C, 2004,1]: "**Loose Coupling**" / W3C /

<http://www.w3.org/2003/glossary/keyword/All/?keywords=loose%20coupling> / Last access 22/07-2008

[Wikipedia]: "**Chinese whispers**" / Wikipedia /
http://en.wikipedia.org/wiki/Chinese_whispers / Last access 22/09-2008

[WfMC]: "**XPDL Implementations**" / Workflow Management Coalition /
<http://www.wfmc.org/xpdl-implementations.html> / Last access 22/20-2008

[ZIFA]: "**The Zachman Institute for Framework Advancement**" / The Zachman Institute

[Zimmermann, Krogdahl & Gee]: "**Elements of Service-Oriented Analysis and Design**" /
Olaf Zimmermann, Pal Krogdahl & Clive Gee /
<http://www.ibm.com/developerworks/webservices/library/ws-soad1> / Last access 24/08-2008

Articles, Essays & Whitepapers & Reports

[BEA]: "**The State of the BPM Market - Business and IT: Solving Process Problems Together**" BEA Whitepapers / 2008 / BEA Systems Inc

[Bézivin, 2005]: "**On the Unification Power of Models**" / Jean Bézivin / 2005 / Software and System Modeling (SoSym) 4(2)

[Blanc, Bouzitouna & Gervais]: "**A Critical Analysis of MDA Standards through an Implementation: the *ModFact* Tool**" / Xavier Blanc, Salim Bouzitouna & Marie-Pierre Gervais / 2006 / Proceedings of the 2006 ACM symposium on Applied computing

[Brahe & Bordbar]: "**A Pattern-based Approach to Business Process Modeling and Implementation in Web Services**" / Steen Brahe & Behzad Bordbar / 2007 / Service-Oriented Computing ICSOC 2006, Volume 4652/2007

[Byrd, Lewis & Bryan]: "**The leveraging influence of strategic alignment on IT investment: An empirical examination**" / Terry Anthony Byrd Bruce, R. Lewis, & Robert W. Bryan / 2006 / Information & Management 43 (2006) 308–321

[Carvallo & Franch]: "**Extending the ISO/IEC 9126 quality model with non technical factors for COTS components selection**" / Juan Pablo Carvallo & Xavier Franch / 2006 / Proceedings of the 2006 International workshop on Software quality

[Chan & Reich]: "**IT alignment: an annotated bibliography**" / Yolande E Chan & Blaize Horner Reich / 2007 / Journal of Information Technology (2007) 22

[Chan et al.]: "**Business Strategic Orientation, Information System Strategic Orientation, and Strategic Alignment**", Yolande E. Chan, Sid Huff, Duncan Copeland & Donald W. Barclay / 1997 / Information Systems Research / Volume 8, Issue 2, June 1997

[Chan, Huff & Copeland]: "**Assessing realized information systems strategy**" / Yolande E. Chan Sid L. Huff & Duncan G. Copeland / 1998 / Journal of Strategic Information Systems 6 (1998) 273-298

[Dico]: "**Delivering SOA with TOGAF**" / Awel Dico / 2008 / The Open Group

[Doucet et. al]: "**Coherency Management: Using Enterprise Architecture for Alignment, Agility, and Assurance**" / Gary Doucet, John Götze, Pallab Saha & Scott Bernard / 2008 / Journal of Enterprise Architecture, May 2008 / Association of Enterprise Architects

[Dursum & Perakath]: "**Towards a truly integrated enterprise modeling and analysis environment**" / Delen Dursum & Benjamin C. Perakath / 2003 / Computer in Industry, Vol. 51

[Erche, Wagner & Hein]: "**Mapping Visual Notatianons to MOF Compliant Models with QVT Relations**" / Michael Erche, Michael Wagner & Christian Hein / 2007 / Proceedings of the 2007 ACM symposium on Applied computing

[Emig et al.]: "**Model-Driven Development of SOA Services**" / Christian Emig, Karsten Krutz, Stefan Link, Christof Momm & Sebastian Abeck / 2007 / Forschungsbericht

[Fitzgerald & Murphy]: "**Business process reengineering: Putting theory into practice**" / Brian Fitzgerald & Ciaran Murphy / 1996 / INFOR / Feb 1996

[Fowler, 2005]: "**The New Methodology**" / Martin Fowler / 2005 / (<http://martinfowler.com/articles/newMethodology.ht9ml>)

[Gartner, 2006]: "**Defining, Cultivating and Measuring Enterprise Agility**" / David W. McCoy & Daryl C. Plummer / 2006 / Gartner

[Gómez-Pérez & Corcho]: "**Ontology Languages for the Semantic Web**" / Asunción Gómez-Pérez and Oscar Corcho / 2002 / IEEE Intelligent Systems, Volume 17, Issue 1, Jan/Feb 2002

[Gruber]: "**A Translation Approach to Portable Ontology Specifications**" / Thomas R. Gruber / 1993 / Knowledge Acquisition, 5(2):199-220, 1993

[Harmon & Wolf]: "**Business Process Management and Service Oriented Architecture**" / Paul Harmon & Celia Wolf / 2007 / BPTrends

[Heflin & Hendler]: "**A Portrait of the Semantic Web in Action**" / Jeff Heflin & James Hendler / 2001 / IEEE Intelligent Systems 16(2), 2001

[Huang & Nof]: "**Enterprise agility: a view from the PRISM lab**" Chin-Yin Huang & Shimon Y. Nof / 1999 / International Journal of Agile Management Systems, Volume 1, Number 1

[Hunton et al.]: **"Enterprise Resource Planning Systems: Comparing Firm Performance of Adopters and Non-adopters"** / James E. Hunton, Barbara Lippincottb & Jacqueline L. Reckb / 2003 / Journal of Accounting Information Systems vol. 4, 2003

[Jeston & Nelis, 2008, 2]: **"Re-inventing Business Process Management"** / John Jeston & Jason Nelis / 2008 / BPTrends

[Kellen]: **"Strategic Agility: Beyond IT"** / Vince Kellen / 2006 / Business-IT Strategies Vol. 9, No. 3 / Cutter Consortium

[Kamoun]: **"The Convergence of Business Process Management and Service Oriented Architecture"** / Faouzi Kamoun / 2007 Association of Computing Machinery, Volume 8 , Issue 24 (June 19, 2007 - June 25, 2007)

[Kleppe]: **"A Language Description is More than a Metamodel"** / Anneke Kleppe / 2007 / Fourth International Workshop on Software Language Engineering, 1 Oct 2007

[Lublinsky & Tyomkin]: **"Dissecting Service-Oriented Architectures"** / Boris Lublinsky & Dmitry Tyomkin / 2003 / Business Integration Journal, October 2003

[Henderson & Venkatraman]: **"Strategic alignment: leveraging information technology for transforming organizations"** / John C. Henderson & N. Venkatraman / 1993 / IBM Systems Journal, March, 1993

[Jenz]: **"Business Process Ontologies: Speeding Up Business Process Implementation"** / Dieter E. Jenz / 2003

[Lundell et al.]: **"UML Model Interchange in Heterogeneous Tool Environments: An Analysis of Adoptions of XMI 2"** / Björn Lundell, Brian Lings, Anna Persson & Anders Mattsson / 2006 / Model Driven Engineering Languages and Systems, Volume 4199/2006

[Joergensen]: **"You can serialize BPMN into BPEL, but BPEL should not be the serialization standard"** / Jesper Joergensen / <http://jesperatwork.blogspot.com/2008/04/you-can-serialize-bpmn-into-bpel-but.html> / Last access 22/10-2008

[Khoshafian, 2005]: **"Narrowing the Semantic Gap between Business Process Analysis and Business Process Execution"** / 2005 / Setrag Khoshafian / Workflow Management Handbook 2005 / Edited by Layna Fisher / Workflow Management Coalition

[McCabe]: **"What's up at OMG"** / Francis McCabe / <http://frankmccabe.wordpress.com/2006/07/01/whats-up-at-omg> / Last access 22/07-2008

[Missikoff]: **"Harmonise: An Ontology-Based Approach for Semantic Interoperability"** / Michele Missikoff / 2002 / ERCIM News No. 51, October 2002

[OEM]: **"Vækst gennem Globalisering"** / Økonomi & Erhvervsministeriet / 2003 / Økonomi & Erhvervsministeriet

[OMG, 2003]: "**MDA Guide V1.0 .1**" / Object Management Group / 2003

[Palmer]: "**Understanding the BPMN-XPDL-BPEL Value Chain**" / Nathaniel Palmer / 2006 / Business Integration Journal November/December 2006

[Parreiras, Staab & Winter]: "**Marrying Ontological and Metamodelling Technical Spaces**" / Fernando Silvas Parreiras, Steffen Staab & Andreas Winter / 2007 / ESEC/ACM FSE-2007 – Proceedings of the 6th joint meeting of the European software engineering conference and the 14th ACM SIGSOFT symposium on Foundations of software engineering, September 03 - 07, 2007, Dubrovnik, Croatia. ACM 2007, pp. 439 – 448

[Provost]: "**On the Cusp: A Global review of the Semantic Web Industry**" / David Provost / 2008 /

[Rambøll]: "**IT i praksis – Executive Summary**" / Rambøll Management / 2003/ Rambøll Management

[Recker & Mendling]: "**On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages**" / Jan Recker & Jan Mendling / 2006 / Proceedings 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortiums

[Sheth]: "**Changing focus on interoperability in information systems: From system, syntax, structure to semantics**" / Amit P. Sheth / 1998 / Conference on Interoperating Geographic Information Systems

[Snowden]: "**Complex Acts of Knowing: Paradox and Descriptive Self-awareness**" / Dave Snowden, 2002 / Journal of Knowledge Management - Vol 6, No. 2, 2002 (May)

[Sousa, Pereira & Marques]: "**Enterprise Architecture Alignment Heuristics**" / Pedro Sousa, Carlos Marques Pereira & José Alves Marques / 2005 / Microsoft Architect Journal, Journal 4

[Sprott]: "**SOA: An Introduction for Managers**" / David Sprott / 2004 / CBDI Forum / (<http://www-1.ibm.com/services/us/bcs/pdf/soa-cbdi-report-2004-july.pdf>)

[Sundblad & Sundblad]: "**Serviceorienterad arkitektur—En översikt**" / Sten Sundblad & Per Sundblad / 2004 [Swenson]: "**A Format for a Process Design Ecosystem**" / Keith Swenson / 2007 / BPTrends January 2007vi

[Terrasse et al.]: "**Do we need metamodels AND ontologies for engineering platforms?**" / Marie-Noëlle Terrasse, Marinette Savonnet, Eric Leclercq, Thierry Grison & George Becker / 2006 / Proceedings of the 2006 international workshop on Global integrated model management

[TOGAF, 2006]: "**SOA/TOGAF Practical Guide Project**" / The Open Group / 2006 / The Open Group

[Uschold, Jasper & Clark]: "**Three Approaches for Knowledge Sharing: A Comparative Analysis**" / Mike Uschold, Rob Jasper & Peter Clark / 1999 / Proc 12th Workshop on Knowledge Acquisition, Modeling, and Management (KAW'99)

[Vdovjak & Houben]: "**RDF Based Architecture for Semantic Integration of Heterogeneous Information Sources**" / Richard Vdovjak & Gert-Jan Houben / 2001 / International Workshop on Information Integration on the Web, ed. E. Simon, A. Tanaka, Proceedings of the WIIW'2001

[Zachman, 1987]: "**A framework for information systems architecture**" / John A. Zachman / 1987 / IBM Systems Journal, VOL 26. NO 3, International Business Machines Corporation

[Zachman, 2000]: "**Architecture Artifacts Vs Application Development Artifacts**" / John A. Zachman, 2000 / Zachman International

[Zachman & Sowa]: "**Extending and Formalizing the Framework for Information Systems Architecture**" / John A. Zachman & John F. Sowa / 1992 / IBM Systems Journal, Vol 31, No 3

Standards

[BPEL]: "**Web Services Business Process Execution Language Version 2.0**" / OASIS / 2007

[BPMN, 2006]: "**Business Process Modelling Notation 2.0 RFP**" / Object Management Group / 2006

[BPMN, 2008]: "**Business Process Modelling Notation 1.1**" / Object Management Group / 2008

[IEEE1471]: "**Recommended Practice for Architecture Description of Software-Intensive Systems**" / IEEE / 2000

[MOF]: "**Meta Object Facility (MOF) Core Specification**" / Object Management Group / 2006 / OMG Available Specification Version 2.0

[OSM]: "**Organizational Structure Metamodel**" / Object Management Group / 2004 / Request For Proposal

[OWL]: "**OWL Web Ontology Language Guide**" / W3C / 2003

[QVT]: "**MOF QVT**" / The Open Group / 2007 / Final Adopted Specification

[SBVR]: "**Semantics of Business Vocabulary**" / Object Management Group / 2008 /
OMG Available Specification Version 1.0

[SOA-RM]: "**Reference Model for Service Oriented Architecture 1.0**" / OASIS / 2006

[TOGAF]: "**The Open Group Architecture Framework**" / The Open Group / 2007 /
Version 8.1.1, Enterprise Edition

[UPMS]: "**UML Profile and Metamodel for Services RFP**" / The Open Group / 2006 /
Work in Progress

[WSDL]: "**Web Services Description Language (WSDL) 1.1**" / W3C / 2001

[XSD]: "**XML Schema definition 1.0**" / W3C / 2001

Encyclopaedia

[Britannica]: "**Architecture**" / Encyclopaedia Britannica / 22/08-2008 /
(<http://www.britannica.com/EBchecked/topic/32876/architecture>)

Presentations

[Beckett]: "**Introduction to RDF Query with SPARQL**" / Dave Beckett / 2006 / Yahoo Inc.