

Service Oriented Enterprise Architecture

- Censored edition

Master thesis at: IT-University of Copenhagen
Author: Rasmus Knippel
Email: rasmus@knippel.org
Supervisor: John Gøtze
November 2005

Abstract

SOA (Service Oriented Architecture) has been an unavoidable acronym for the last couple of years, and it is exactly because it has been around for some time that I find it so interesting. Behind the acronym and sales pitches, an enormous evolution has taken place. An evolution, not so much focused on technology, but on the paradigm of Service Oriented Architecture.

This thesis has the advantages of this evolution and history of SOA, but this also presents a challenge as the concept has no clear definition. The first task of this thesis is to clarify how the key concepts of this thesis are to be perceived.

Evolution is also the driver for the rest of this thesis, as the evolution of SOA has entailed that it is now operating on a much higher level of abstraction - a level of abstraction similar to where Enterprise Architecture (EA) operates. Having two architectural drivers operating on the same level of abstraction is bound to create conflicts.

This thesis seeks out to identify the areas where SOA and EA correlate. This is done by approaching the issue from different perspectives. From which the most important finding is identified using Peter Herzum's article "Applying Enterprise Architecture" [57]. Applying SOA to this Maturity Model shows that the definition of SOA equals the definition of EA at its most mature state - called Nirvana.

The paths of SOA and EA to this state of Nirvana are not identical, but by aligning these it is possible to get a synergy that will create a much stronger coupling between the business and IT - creating the agile business. SOA is however not the "silver bullet", but aligning EA and SOA provides the ability to get a holistic view of the entire enterprise. Aligning SOA and EA will change both SOA and EA in a degree resulting in a new concept: Service Oriented Enterprise Architecture (SOEA).

The thesis sets the base for development of SOEA. The base is founded in a conceptual alignment of concepts of SOA and EA. However, as a big part of the motivation behind this thesis is to move from the conceptual level, concrete examples of where SOA and EA correlate was identifying. This is done through the use of SOA Artifacts, which are mapped against EA Artifacts. This "exercise" showed that SOA changes EA at a very high level of abstraction and, add new aspects to EA at a lower level of abstraction.

The way this thesis differs from other work to better connect SOA and EA, is that it is not trying to integrate the two directly. The overall belief is that the both SOA and EA will be subject to fundamental change, and it has been seen as an important task to identify why this is necessary. The answer was found in the spurious correlation between SOA and EA: the fact that SOA and EA operate on a similar level of abstraction can be used to get a strong coupling between the two. It is not a choice of one over the other; it is a matter of integrating the two, and getting the best capabilities of both.

Contents

ABSTRACT	II
CONTENTS	I
FIGURE LIST	III
1 TARGET AUDIENCE AND PREREQUISITES	1
2 MOTIVATION	2
3 THESIS PROBLEM	4
4 METHOD	5
4.1 THESIS STRUCTURE.....	6
5 THE HYPE OF SOA	7
5.1 THE SUCCESS OF SOA.....	9
6 BACKGROUND AND CONCEPTS	11
6.1 DEFINING CONCEPTS.....	12
6.1.1 Enterprise Architecture.....	13
6.1.2 Enterprise Architecture Process.....	14
6.1.3 Enterprise Architecture Framework.....	15
6.1.4 Zachman Framework.....	16
6.2 SERVICE ORIENTED ARCHITECTURE (SOA).....	17
6.2.1 SOA and Web Services.....	18
6.2.2 The SOA Process.....	18
6.2.3 Services.....	20
6.2.4 Web Services.....	21
6.2.5 Loosely coupled systems.....	23
6.2.6 Web Service Levels.....	25
6.3 SUMMARY.....	28
7 EVOLUTION AND MATURITY OF SOA AND EA	29
7.1 SOA AND SOFTWARE DEVELOPMENT.....	29
7.2 EVOLUTION IS CONSTANT.....	31
7.3 THE EVOLUTIONARY PHASES OF SOA.....	34
SOA PHASE I	34
SOA PHASE II	35
SOA PHASE III	35
SOA PRE-PHASE	35
7.3.1 SOA Phase IV.....	36
7.3.2 The EA phase of SOA.....	36
7.4 PARALLELS OF SOA AND EA.....	37
7.5 MATURITY OF SOA AND EA.....	39
7.6 SUMMARY.....	44
8 PERSPECTIVES OF SOA'S IMPACT ON EA	46
8.1 BUSINESS PERSPECTIVE.....	46
8.2 APPLICATION PERSPECTIVE.....	48
8.3 INFORMATION PERSPECTIVE.....	48
8.4 TECHNOLOGY PERSPECTIVE.....	50
8.5 SUMMARY.....	50
9 SERVICE ORIENTED ENTERPRISE ARCHITECTURE	52
SOA - A + EA = SOEA	52
9.1 SOA ARTIFACTS.....	53
9.2 IDENTIFYING SOA ARTIFACTS.....	54
9.2.1 Types of SOA Artifacts.....	71



9.3 SOA ARTIFACTS VS. EA ARTIFACTS	75
9.4 SOA ARTIFACT RELATIONS	78
9.5 IN-DEPTH ANALYSIS OF TWO SOA ARTIFACTS	80
9.5.1 <i>Enterprise Service Bus Policy</i>	80
9.5.2 <i>Version Control</i>	84
9.6 SUMMARY	87
10 REFLECTION AND PERSPECTIVE.....	89
10.1 CHAPTER SUMMARY	89
10.2 THE EVOLUTION	90
10.3 SERVICE ORIENTED ENTERPRISE ARCHITECTURE.....	93
10.4 SOEA PROCESS	95
10.5 SOEA GOVERNANCE.....	96
10.6 SUMMARY	97
11 CONCLUSION	99
12 AFTERMATH	100
13 APPENDIX.....	101
13.1 APPENDIX A	101
13.2 APPENDIX B.....	104
13.3 APPENDIX C.....	106
13.4 APPENDIX D	109
13.5 APPENDIX E.....	110
13.6 APPENDIX F	112
14 BIBLIOGRAPHY	113

Figure list

Figure 1 : Ungoverned, unmanaged SOA introduces challenges [Source 106].....	3
Figure 2 : Thesis structure	6
Figure 3 : SOA Hype Cycle as of April 2004 [Source: 59, slide 15]	9
Figure 4: Elements of an Enterprise Architecture [Source 7 p.15].....	15
Figure 5: The Zachman Framework [Source: 29]	16
Figure 6: SOA, CBD and OO in Context [Source: 40].....	21
Figure 7: Gartner Web Service Hype Cycle [Source: 130]	22
Figure 8: Loosely- vs. Closely coupled systems [Source: 8, s. 133].....	25
Figure 9: Service Levels [Source: 20, p11]	26
Figure 10: Service ontology [Source : 46].....	27
Figure 11: SOA, CBD and OO in Context [source: 40]	30
Figure 12: The Conventional Technology S-curve [Source: 58 p. 40]	30
Figure 13: Evolution of paradigms [Source: own work].....	31
Figure 14 : The Landscape of the Adoption Life Cycle [Source: 75 p. 25].....	31
Figure 15 : High-Tech Sector Growth Model [Source: 75. p. 105]	33
Figure 16 : SOA Phases [Source: own work].....	36
Figure 17 : Typical EA maturity phases. [Source: 57, p. 8].....	40
Figure 18 : SOA Maturity Model Levels with Key Business Impact [Source: 129].....	45
Figure 19 : Views of Enterprise Architecture [Source: 89, p. 54]	46
Figure 20 : Defining Business Services [Source: 89, p. 56].....	47
Figure 21 : Typical SOA Application Architecture [Source: 89, p 57].....	48
Figure 22 : Normal information view [Source: own work].....	49
Figure 23 : SOA information view [Source: own work].....	49
Figure 24 : Conceptual View of Technology Architecture [Source: 89, p 57].....	50
Figure 25: If/then view of architecture [Source : 65]	57
Figure 26 : SOA Governance [Source: 104]	63
Figure 27: SOA peer to peer [Source: 31]	67
Figure 28: The Enterprise Service Bus [Source: 2].....	68
Figure 29 : SA-9: Web Application Diagram [Source: 110].....	77
Figure 30 : Relations between SOA Artifacts [Source: own work]	79
Figure 31 : From business needs to ESB [Source: 112].....	81
Figure 32: Relations to ESB Policy [Source: own work]	81
Figure 33: EA D-4 and SA-9 mapping [Source: own work].....	83
Figure 34: ESB-policy EA - SOA mapping [Source: own work]	84
Figure 35 : The Beer Game of SOA [Source: own work]	85
Figure 36 : Placing of SP-3: System Accreditation Document [Source: 110, p.40].....	86
Figure 37: Version Control (Change Control) - SOA mapping [Source: own work]	87
Figure 38 : Crossover of EA and SOA [Source: own work].....	91
Figure 39: The link between EA, SOA and Interoperability [Source: 126 - translated from danish].....	91
Figure 40 : The link between EA, SOA and Interoperability - updated version [Source: own work].....	92
Figure 41 : The steps of Nykredits's SOA project [Source: own work].....	92
Figure 42: Service Oriented Enterprise Architecture [source: own work]	93
Figure 43 : The "road" to Nirvana [Source: own work].....	94
Figure 44 : The "SOEA-road" to Nirvana [Source: own work].....	94
Figure 45 : Abstraction and maturity of EA and SOA [Source: own work]	95
Figure 46 : Abstraction and maturity of SOEA [Source: own work].....	96
Figure 47 : SOEA - the alignment of SOA and EA [Source: own work]	96
Figure 48 : The link between EA, SOA and Interoperability.....	98
Figure 49 : SOEA - the alignment of SOA and EA [Source: own work]	98



1 Target audience and prerequisites

The primary target group of this thesis is of course, as probably all theses: my external examiner and supervisor. Secondly the target group is people generally interested in Service Oriented Architecture (SOA) and/or Enterprise Architecture (EA). Colleagues are also a part of my target audience - after all they are partly responsible for many of my thoughts in the more practical aspects of SOA and EA.

EA and SOA are concepts covering almost all levels of IT, hence it will be impossible for me to describe all the aspects of either of these. My focus is how SOA will affect EA, and my perspective will primarily be from the "the SOA side". However, as I see the correlation between SOA and EA as the interesting issue, the thesis should interest people from both the SOA- and the EA world.

There is a big challenge in looking at both EA and SOA as neither of the concepts are clearly defined, and to fully define the concepts here would be an impossible task within the scope of this thesis. I will explain my views on both concepts, and related concepts, but it is required for the reader to have experience with both SOA and EA.

The reader should be well founded in the discipline of both Enterprise Architecture and Service Oriented Architecture. This means that general knowledge to IT is required such as; software development methods, programming language, Enterprise Resource Planning (ERP) systems, IT Standards, Standardisation organisations and the big vendors on the market.



2 Motivation

Trying to specify a single reason for my choice of subject for this thesis is not an easy task. However I think I have identified the true reason: Frustration, and perhaps even annoyance; In order to explain this I have to take you back a few steps. For the last couple of years I have been working with the concept of Service Oriented Architecture (SOA) as a student in the Danish Ministry of Science Technology and Innovation. It has been an interesting journey where my own understanding of SOA has changed quite a bit since my first encounter with the concept. My perception of SOA has moved from a technical view, to seeing SOA as a part of Enterprise Architecture (EA). This personal "evolution" has furthermore turned my attention towards EA, not only as student at the IT- University, but in professional career as well.

At the end of my work on this thesis the report "OECD Peer Review of e-Government in Denmark" was published with one of the main issues being:

"A major concern that, while the enterprise architecture and supporting standards and frameworks have been very well developed at the conceptual level, they are proving more difficult to translate into the actual standards and schemas required for implementation. Many people working to implement the architecture find it abstract and difficult to understand."

[126, p. 107]

This finding of OECD is a good illustration of my frustrations on the work which has been done by the very office I work at. I have a practical background and have had little luck in identifying the real value of the EA initiatives being done in the Danish government. The work being done definitely has a value, but if it is not brought from the abstract level pointed out above to a more practical level, the values will never be utilized.

One of the choices of the Danish EA initiative has been that of SOA. This choice has unfortunately been made without defining the concept of SOA - and SOA is not just SOA! I believe that in order to work with SOA each enterprise, public or not, must define the concept of SOA as many conflicting definitions exist.

If not taking this challenge seriously I foresee that the result of "going SOA" will end in chaos - as illustrated in the following:

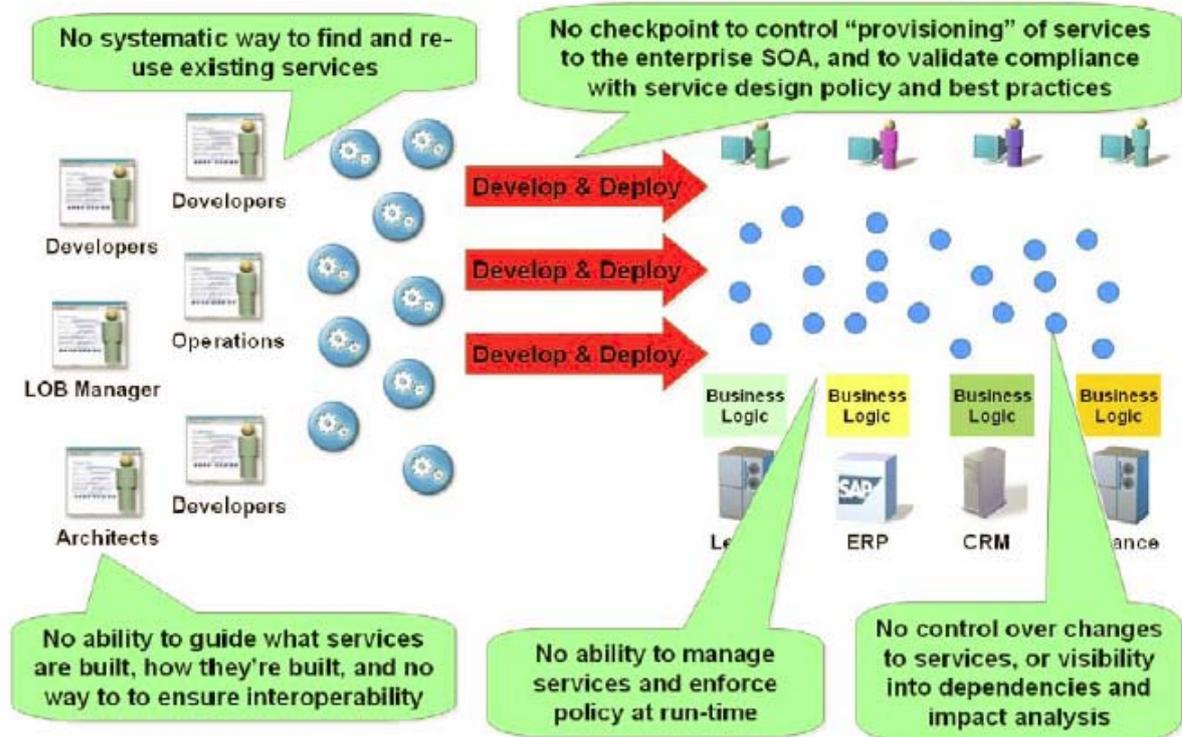


Figure 1 : Ungoverned, unmanaged SOA introduces challenges [Source 106]

The important issue of Figure 1 is that it in fact illustrates that the characteristics you will end up with are similar to those prior to SOA. So, is this an illustration made to intimidate people thinking of SOA? If you only think of SOA as a technical issue I hope it does.

Figure 1 also illustrates the reason for my frustration - this is not SOA to me! It might fit many definitions of SOA, but then again you can also use an object oriented programming language without your code being Object Oriented. SOA is a big challenge which is not merely introducing Services.

I have seen several enterprises claiming that they are using SOA, but in my view they are at most sub-optimal implementations [18, p, abstract]. This is also illustrated by the following quote from Jeff Schneider's blog:

"More CIO's will lose their job over SOA implementations than lost their job over ERP implementations."

[96]

What he is referring to is that this will be due to CIO's underestimating the SOA. I will return to this statement in my thesis.

So, why am I annoyed? Because I in fact see the paradigm of SOA as being very powerful. However before we can harvest the promised benefits of SOA we must take it seriously, and understand how to control SOA on an enterprise-wide level.



3 Thesis Problem

Supporting the business has always been the purpose of IT. However, in an evermore competitive world the ability to change has become more important than ever. Service Oriented Architecture (SOA) has often been proclaimed to be the “silver bullet” that can make your IT flexible and support the desired agility of the business. The following quote illustrates how serious this problem is:

“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.”

[18, p. 3]

SOA is the natural evolution in software development, enabling a higher level of abstraction, which has made it easier to lift IT to a management level. This higher level of abstraction has lead to conflicting areas between Enterprise Architecture (EA) and SOA. A conflict that need to be approached with caution. But, if solving this conflict it is my claim that SOA and EA will form a synergy enabling the harvesting of the promised effects of both SOA and EA.

Before a conflict can be solved it must be known, and in order to resolve the conflict a common ground must be identified. The purpose of this thesis is to set the first step in this direction by:

- Describing the evolution of SOA as a concept.
- Identify, on a conceptual level, the aspects where SOA and EA operate on common levels of abstraction.
- Elaborate on the conceptual aspects, using operational aspects commonly used by both SOA and EA.
- Discuss the next step of SOA and EA.

Problem summary: Does SOA and EA have a spurious correlation¹, or is there something causal between the two?

¹ <http://www.burns.com/wcbspurcorl.htm>



4 Method

The project at hand is divided in two main areas; EA and SOA. The discipline of EA is better founded in the academic literature than SOA, but, as stated in the introduction there is a lot of hype around SOA, and there are no practical examples on businesses that have fully implemented SOA and hardly any academic work on SOA.

SOA is an emerging field in academia, but is mainly described in consulting literature. This has divided my research approach into two different paths:

1. The study on EA is founded on analysis of academic literature, and some consulting literature.
2. The study on SOA is mainly founded in consulting literature, and some academic work.

The reason for using consulting experiences is that academia in general is notoriously behind practitioners, but also with a more critical view! The use of consulting literature must be done with care and with a critical mind.

The thesis will be written using an explorative approach based on my personal experience on the subject. This means that the theoretically base is founded on one of the main areas of the thesis; SOA.

I have chosen not to use a case driven approach as my experience of the research area is that there are little practical experience on the area, or perhaps even more important, the experience is often contradictory within the individual enterprise. I experienced an example of this attending a conference on SOA, where I asked Danske Bank's² chief architect; if it wasn't a problem defining the granularity of the Services? His reply was that this was no problem at all, but after the conference I was approached by a developer from the very same company telling me that this in fact was a major issue.

The world I am about to enter in this thesis is filled with contradictory definitions, and to fully argument for which of these are most correct could be the subject of several thesis'. As I have been working with the concepts of this thesis for a couple of years I will not define the concepts of this thesis as such, the knowledge of these are set as a prerequisite to the reader (see "Target audience and prerequisites"), I will however clarify my interpretation of the concepts.

I will during process of writing discuss concepts and related issues with professionals in the business, but these are not to play a dominant role in the content of the thesis. These statements will be a part of a hermeneutic study based on; academic literature, consultancy reports and blogs. The explorative approach using a hermeneutic perspective must entail that arguments are backed by literature references, but also that there an interpretation of these.

² Danske Bank is the largest bank in Denmark and a leading player in the Scandinavian financial markets (<http://www.danskebank.com/About>).

4.1 Thesis Structure

The thesis is structured in four main parts followed by a conclusion, as shown in the following graphical illustration.

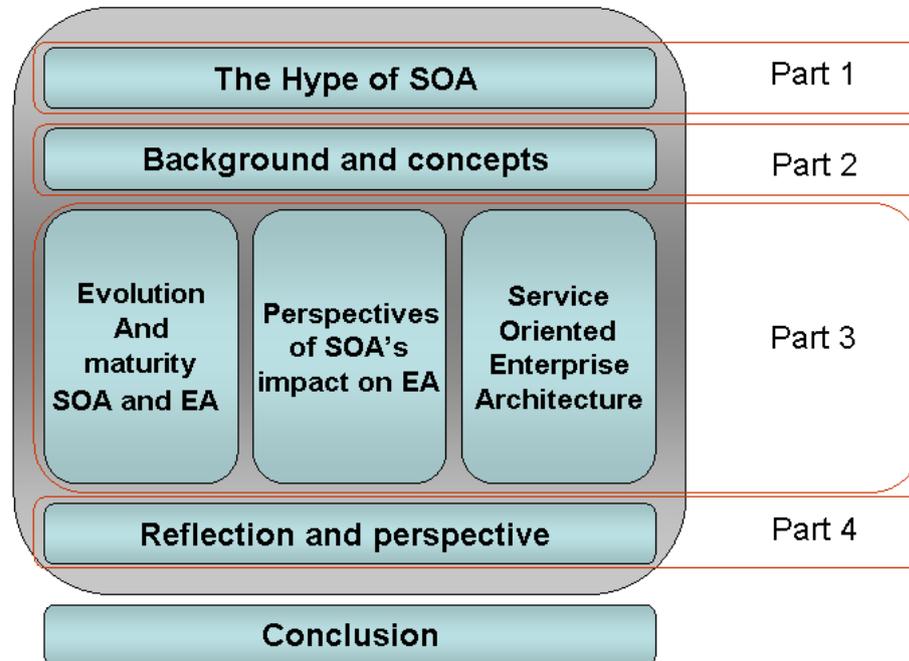


Figure 2 : Thesis structure

Part 1 is a short introduction to the promises of SOA. The motivation of this part is to serve as an appetizer, but also to illustrate that how SOA is sold as a concept.

Part 2 is to set the theoretical base for the rest of the thesis. It is important to note that the focus is to clarify how the concepts of this thesis are to be perceived, and not an in depth description of the general definition of the concepts.

Part 3 consists of three chapters. The focus of these three chapters is to approach the problem of the thesis by using different perspectives. Although these three chapters are illustrated as three parallel chapters they are to be seen as a collective whole.

Part 4 is based on the previous chapters of part three. The purpose is to use the different perspectives and identify whether SOA and EA have a spurious correlation, or if there is something causal between the two.



5 The Hype of SOA

When introduced to SOA it is often not the concept of SOA that gets the main attention, but instead what can be gained from choosing SOA that steals the picture. This is in my opinion an unfortunate development, founded in marketing efforts. I will return to defining SOA, but first I will take a look at some of the promises or sales pitches of SOA:

- **Agile systems** [40] [54][15, p.5]
SOA will enable you to easy and swift change your system. Not only in terms of functionality but also; geographical placing of systems, change of platform, changing vendor etc.
- **Easy integration with both inside and outside partners - cutting expense** [54] [18, p. 6] [2, p. 34] [40] [39].
The ability to seamlessly integrate systems across platform and silos is by some seen as what SOA is all about [39].
- **Reuse** [55] [54] [18, p 3] [15, p.5]
Reuse of code and systems have been the main driver behind the development of new programming paradigms and development methods. SOA promises to deliver reuse not only in terms of functionality but also data through the principle; data once available everywhere.
- **Supporting “short lifetime” products** [24] [18, p. 4].
Reduced Time To Market (TTM) for new Services/products.
Competition in the world is increasing, and the life-time of products is decreasing. SOA promises the ability to fast support new products.
When expanding the number of Services of an SOA a lot of the work has been done and can be reused. Which again entails short development cycles [2, p. 34].
- **Improving Return on Investment (ROI)** [24] [2, p. 34]
Reduced Total Cost of Ownership (TCO) of IT infrastructure and business Services [2, p. 34] by:
 - Eliminating costly, proprietary middleware and replacing it with equally capable, open standard-based Web Services technologies [24]
 - Consolidating well-defined business functions into Services that can be shared by multiple business units [24]
- **Mapping of business processes directly into IT**
Integration between business and IT is the key design-rule of SOA - all processes in the business must be thought of as Services. This needs support from the managerial level of the business, and SOA is easy to explain to managerial level - high level of abstraction [18, p. 4].
- **Incremental implementation** [15, p.5]
SOA is not a “big bang” system. SOA is an evolution of the existing systems by breaking down barriers between existing systems [18, p. 4].
- **Creating real-time/on demand systems** [18, p. 4].
The concept of the; “On Demand Business” is a concept trademarked by IBM [60]. It is a parallel to the previous “Supporting short lifetime products” and the same comments apply here.



- **Flexibility to interchangeably change from one Service provider to another** [18, p. 4].
Integration is not only an internal issue.
- **Cut down in development cost** [55]
The development cost will initially be higher, but will when the SOA matures be lower.
- **Identifying Services can help the business identifying the core businesses** [15, p.5].
The process of defining the processes of the business will be good for analyzing the business.
- **Concepts of the business must be defined in a unanimous way.**
SOA requires a common data model.
- **Information architecture becomes visible** [15, p.5].
The process of specifying and maintaining the common data model will be visible to the entire business.
- **Using standards ensures interoperability.**
Using standards have always been the way to ensure interoperability. The new step is to lift it from being a technical issue to the business [61].
- **Platform independence** [18, p. 4].
SOA can work across multiple platforms through standard interfaces.
- **Place independent**
Physical location can be changed runtime.
- **Improve data quality**
It is possible to Improve the quality of data, because data is distributed and not replicated, thereby creating more precise reports and data analysis.
- **Clear definition of responsibility of Services improves ability to make departments accountable.**
Service Level Agreements (SLA) on all Services defines responsibility.
- **Making IT Governance easier** [18, p. 4].
Easing central overview of the IT Enterprise [2, p. 34] [15, p.5].
- **First mover hype - media coverage** [59, slide 15]
Possible to market the company as an innovative organisation.

All these advantages, which is only a subset of what is proclaimed about SOA in the marketing efforts, makes it hard not to have a closer look at SOA. This has definitely been the case in almost every consultancy house such as Gartner, the source of the following figure.

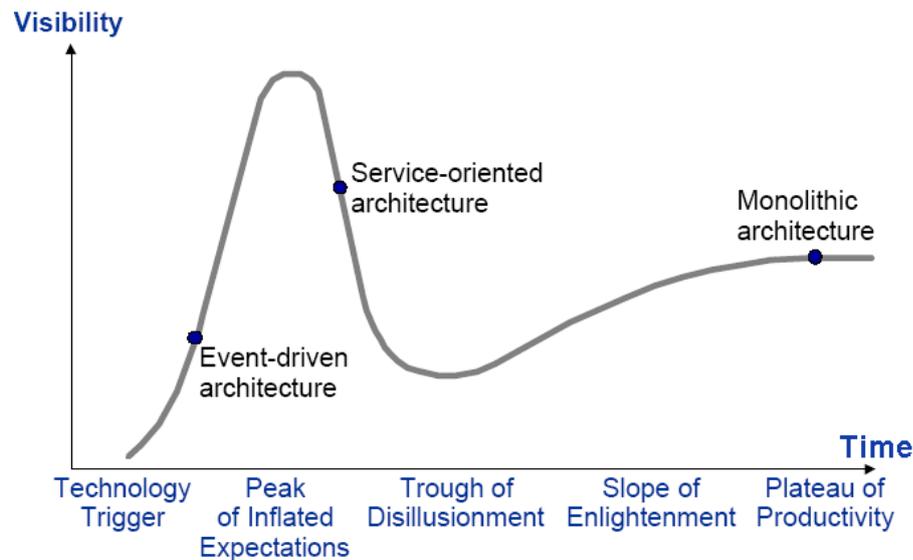


Figure 3 : SOA Hype Cycle as of April 2004 [Source: 59, slide 15]

However Figure 3 shows SOA on the way down after peaking on the Hype Cycle, and since the figure is more that a year old it should be even further down the Hype Cycle. That SOA is going down the Hype Cycle slope fits very well with my view on SOA. SOA started out as being a technical miracle that could solve all the emerging problems that started to arise without the traditional vender login and inflexibility of the existing monolithic systems.

After the initial hype, and practical experiences started to emerge, SOA as a concept has evolved to not only being seen as a technical issue. Services might solve some of the problems of the monolithic systems, but new issues that need focus on where introduced - making everything as Services is not SOA.

5.1 The success of SOA

SOA is on the agenda of almost every larger enterprise working seriously with IT. I have tried to ask myself why SOA have been such a big success, even though no one really know what SOA is - hence no one has proven that SOA works. As discussed in the previous chapter many people think they know what SOA is, including myself, but as long as there is no common agreement on the understanding of the concept SOA it is difficult even to discuss SOA (see section 6.2).

As pointed out in section 5 there are almost no limits to what SOA eligibly can do for your enterprise, which is one of the reasons for the success of the concept of SOA. However, I believe the real reason for the success for SOA is founded in the simple fact that; SOA can be "sold off" as useful to all levels of an enterprise. I will shortly illustrate this point by giving some examples of the obvious advantages of SOA seen from different levels:

- **The Chief Executive Officer (CEO)**
New products or processes will be easy to implement in IT. The possibility to get agile systems that will not diminish the business' opportunity to develop new products fast and seamlessly.
- **The Chief Information Officer (CIO)**
The nightmare of integration will disappear. The system silos will no longer exist. As a result I can easily meet the demands of the business.



- **The Project manager**
The projects can easily be split up into smaller projects which can be solved and implemented independently - making it easier to control the progress of the project.
- **The Developer**
Integration to other systems is probably the most boring task for a developer. Since Web Services currently are the dominant way to implement Services this can be done very easily though the tool support of all major platforms.
- **The User**
You will need only one system - no more manually integration, such as copy paste by the users between the systems. The complexity that exist beneath the hood will be hidden.

The points above are examples of why the different levels of an enterprise will view SOA positively, many more reasons that could be pointed out. The key issue here is that it is a way of ensuring that all levels of the enterprise benefit from SOA, and therefore you are likely to limit the reluctance that often will come from change.



6 Background and concepts

I have chosen to include a reference to the work of Robert L. Glass et. al. [127] is that the topic of this thesis will cross the boundaries of the three fields of computer disciplines. I will be working on a background that might differ from the reader which can create misunderstandings that can stem from the above mentioned points.

When working with IT one often meets terms that are defined in several ways. If we don't look at direct conflicting definitions of concepts, I believe that the main reason lies in the different levels of abstraction on which we work and look at IT, and our different backgrounds.

Robert L. Glass et. al. has looked on how and what the fields³ of IT focus their research;

- what are the topics of interest,
- how is the problem approached,
- which methods are used,
- what is their frame of reference and,
- what is the level of abstraction.

I have summarized the key findings of the work by Robert L. Glass et. al:

- The different fields have singled out a set of topics on which to focus its research, topic areas that have little overlap.
- Preferred research approaches and research methods do not necessarily command the respect of the other disciplines.
- Researchers have not, in the past, communicated well with each other across the boundaries of their field of interest.
- Terminology differs, sometimes in important ways across fields
- And what may be the biggest problem of all, there is a tendency for each of the fields to disdain the work of the others.

To avoid misinterpretations of concepts I will define the concepts where I consider it necessary. Since it is impossible to define all concepts, some will be based on my background which is in Computer Science (CS) and Software Engineering (SE).

"I shall not today attempt further to define pornography... but I know it when I see it"
[Justice Potter Stewart, US. Supreme Court]

³ Computer science (CS), software engineering (SE), and information systems (IS).



6.1 Defining Concepts

Using concepts derived in the disciplines of computers is often a source of misunderstanding, as described in section 6. To avoid this I will start of by defining my view of the most central concepts used in SOA. I will not describe the concepts in depth, but to a level where it is possible to identify my conception of the given concept.

I have chosen to use a Top Down approach in describing the concepts, hence not start of by defining SOA but instead start with Enterprise architecture (EA). The reason I introduce EA is founded in my motivation for my thesis (see section 2). I see some clear dependencies between EA and SOA; to give an example I will quote David Sprott from CBDI⁴:

“Who is responsible for delivering business adaptability? Does this question even get asked? In my experience most business people think that the obligation to deliver business adaptability is simply an IT issue. The IT industry reinforces that idea by making adaptability an IT architectural issue. If you have loose coupled, Web Service based processes you must be able to respond rapidly to change. Of course regular readers of CBDI all know different, BUT I know lots of business people that are utterly convinced that adaptability is NOTHING to do with them - it’s someone else’s problem.”

[62]

This quote from his article “*SOA IS A BUSINESS ISSUE*” illustrates that you have to tie SOA closely to the Business [62]. EA has until now been the tool for tying the Business and IT together, but by combining EA and SOA I see an even bigger potential to have “Business IT”, than using the two on their own.

⁴ CBDI Forum is an independent industry analyst and consultancy company (www.cbdiforum.com)



6.1.1 Enterprise Architecture

EA as a formal concept is commonly agreed to originate from the “*Zachman Framework for Enterprise Architecture*” developed by John Zachman. The development of the concept started to take form in his article “A framework for information systems architecture” from 1987 [65]. Therefore John Zachman is often referred to as The “Father” of EA.

EA has strong ties into the world of business, which is also indicated by the “E” of EA, and the father of EA is a fellow for the collage of Business Administration at the University of North Texas [3, p. 638] - EA is the business view on IT, and should manage IT as a business [57, p. 2]. EA is a non-technical discipline that should address the strategic and architectural aspects of IT in the business [57, p. 2].

The definition of EA by Philip Allega is the definition I will be working with in this thesis:

“IEEE 1471 defines architecture as “the fundamental organization of a system embodied by its components, their relationships to each other and to the environment, and the principles guiding its design and evolution” (see IEEE P1471/D5.3). EA moves beyond this definition to embrace an aggregated, holistic view of all systems, people, and internal and external constructs that have relationships within the enterprise. Furthermore, it is bound and guided by a common requirements vision (CRV) and a set of conceptual architecture principles that guide the selection, creation, and implementation of business, information, technology, and solution future states.”

[23]

What is important to note is that the EA results in physical products that documents your EA, but that EA is not a physical product in itself. From the definition of EA it is clear that the result of EA is Components and Artifacts [1, p. 111] or the more general term architecture products⁵ [7, p. 14] - but of what, how and when?

Scott A. Bernard defines EA as two parts; a Management Program and a Documentation Method:

A Management program provides [1, p. 33]:

- Resource Alignment: Resource planning and standards determination
- Standardized Policy: Resource governance and implementation
- Decision Support: Financial control and configuration management
- Resource Oversight: Lifecycle approach to development and management

A Documentation method provides [1, p. 34]:

- EA Approach: A modelling framework and implementation methodology
- Current Architecture: Views of as-is strategies, processes, and resources
- Future Architecture: Views of to-be strategies, processes, and resources
- EA Management Plan: A plan to move from the current to the future EA

The two parts of EA, as defined by Scott A. Bernard, I see as a good description of what EA should provide - e.g. what should be the results of the EA? However, what we still are missing is the how and when - the EA process!

⁵I will use the terms *Artifacts*, *Components* and *architecture products* interchangeably. I will refer to *architecture products* as *EA products* as I am working in a context where the reference to architecture can be placed at many levels.



6.1.1.1 Why EA?

Before looking at the EA Process I will look at the problem which EA is supposed to solve. John Zachman's motivation for developing the Framework was to improve the managing of the increasingly complex Information Systems (IS). The reason for the increased complexity was not only the increasing size of the systems, but also that the systems started to get distributed across the enterprise. This original motivation was founded as a result of the technical evolution of IT, however this motivation has evolved in parallel with the concept of EA.

This evolution has meant that the original "why" has been replaced with the advantages of having an EA. In short the advantages can be formulated as; Better, Faster and Cheaper [66][67]. Or in a bit more detail, the motivation for commencing in the world of EA [68]:

- IT costs too much
- Costs of managing complexity
- Eliminate redundancy
- Growing IT ecosystems
- Demanding rate of change
- Need for info-sharing
- Outsourcing (BPO⁶)
- Future-proofing

This is a clear development that has moved EA from Zachman's technical motivation to being formulated from the consequences of the technical evolution. However, all the motivations behind creating an EA is founded in increasing profits - Whether it by saving by expenses or gaining competitive advantages.

Seen in the competitive world of commerce it was so eloquently put by the Red Queen to Alice in Lewis Carroll's "Through the Looking Glass":

"In this place it takes all the running you can do, to keep in the same place."

[69]

The driver for EA is: "The Red Queen Principle" and what you want from your EA is the ability to "run" without straining yourself.

6.1.2 Enterprise Architecture Process

There is no single answer to how an EA process should be performed - this is a process that must be fitted to the individual organization. Jane Carbone has authored the "*IT Architecture Toolkit*" [5] which sets out to describe, what I see as guidelines, on how to approach the EA Process. I see her toolkit as general guidelines because it must be fitted to the individual organisation.

Scott A. Bernard, in his book "*An Introduction to Enterprise Architecture*" [1], sets out between the practical approach of Carbone and a more academic view of EA.

⁶ Business Process Outsourcing

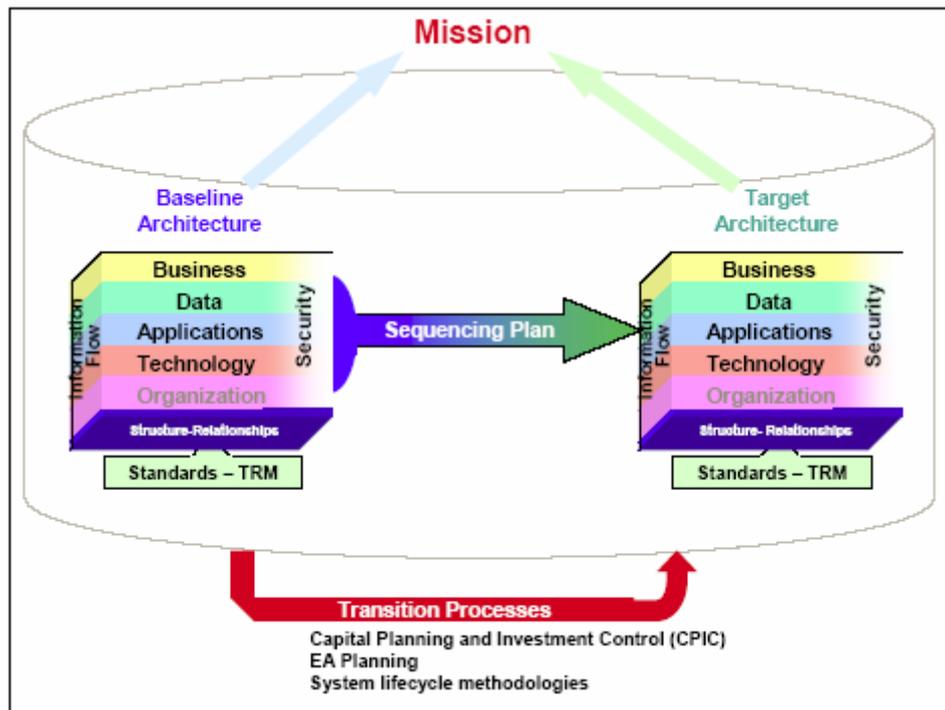


Figure 4: Elements of an Enterprise Architecture [Source 7 p.15]

The usual approach to EA is to describe the *current architecture*⁷ and the *target architecture*⁸ [5][7][1]. Then the EA-process can begin in identifying how to get to the goal: The Transition Process. Defining the current- and target architecture is of course also a part of the EA-process.

6.1.3 Enterprise Architecture Framework

EA and EA Framework are often used interchangeably [7, p 14]. This misconception I see as based on some historical aspects, to which I will return. EA will create a lot of information in the shape of Components and Artifacts [1, p. 111]. This myriad of information needs to be stored in an organised manner, so that retrieval of information is lossless and easy [3, p. 5]. This is the role of the EA Framework, which must [7, p 28]:

- Identify the types of information needed to portray an EA
- Organize the types of information into a logical structure
- Describe the relationships among the information types. Often the information is categorized into architecture models and viewpoints.

These points are similar to how you might describe how you work with relational databases. When storing data in a database, the first task is to identify the relations between the data in the organisation. The reason for comparing the EA Framework is not to make a parallel as such but only to refer to a well known discipline in computer science. Storing the EA products from the EA process is an even bigger challenge than working with relational databases as you seldom know what you are going to store, and even more difficult, is to define the relations between the EA products.

The complexity of defining the relations introduces the need for a more general view on information - a Framework, or more precisely an EA Framework.

⁷ Called "as-is" by Carbone. Also referred to as the "Base Line Architecture" [7]

⁸ Called "to-by" by Carbone

“A framework is a logical structure for classifying and organizing complex information. An enterprise architecture framework provides an organizing structure for the information contained in and describing an EA”

[7, p. 14]

6.1.4 Zachman Framework

As I have previously stated; John Zachman is usually referred to as the father of EA [1]. The main reason for this title is that he is the creator of “A Framework for Information Systems Architecture” in the late 1980’s [7, p. 29]. The concept of “enterprise” was first introduced in the early 1990’s and the Framework is now called; “*Zachman Framework for Enterprise Architecture*” (from here referred to as Zachman Framework).

The change of name is not the only change that has occurred over the years. The Framework itself has also been a subject of change [1, p. 37][7, p. 29].

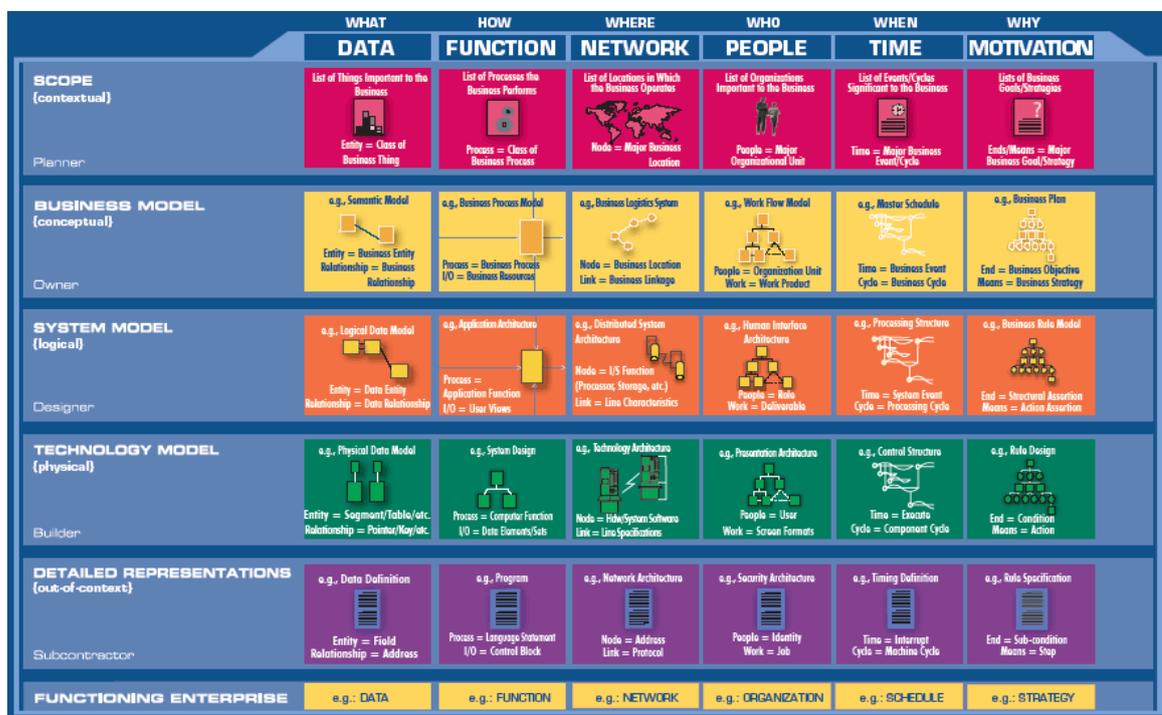


Figure 5: The Zachman Framework [Source: 29]

Today Zachman is viewing his framework, depicted on Figure 5, as a “thinking tool” to help architects and managers through the EA-process [7, p. 30]. This view is also supported by the work of Jane Carbone on her book “*IT Architecture Toolkit*” and Scott A. Bernard on his book “*An Introduction to Enterprise Architecture*”. Their work is based on the Zachman Framework, where they to some extent use the Framework as a focus point to explain their views on the EA-process. And this is also how I see the Zachman Framework; it is not an up to date Framework that can be used in today’s practise, but it is a common frame of reference.

Besides the work of Jane Carbone and Scott A. Bernard there are many other Frameworks that builds on the work of John Zachman, just to name a few [76]:

- The U.S. Department of Defense (DoD) Architecture Framework (DoDAF)
- The Open Group Architecture Framework (TOGAF)
- United States Government Federal Enterprise Architecture (FEA)

I will not further discuss these as this would be out of scope of this thesis.



6.2 Service Oriented Architecture (SOA)

SOA is probably one of the currently most hyped words in IT [32], but generally there are two views on SOA that are distinctively different. The key issue here is whether SOA is a technical issue or not, and it is my belief that the technical definitions often originates from a low-level⁹ perspective on SOA [34].

One of the tricky parts of defining SOA is exactly that it affects almost all levels of IT, from the top level of EA to the specific implementations of Services (see section 5.1). Therefore SOA is often seen defined in a given context or level of abstraction. This complexity has resulted in a very diverse comprehension of SOA.

Ali Arsanjani defines SOA as:

“SOA is not a product - it’s about bridging the gap between business and IT through a set of business-aligned IT services using a set of design principles, patterns and techniques.”

[27]

Interestingly this definition is quite close to the definition of EA. Where it differs, is the “Services”! The Services are a part of the SOA, but implementation of Services does not equal SOA.

CBDI defines SOA as:

“Service Oriented Architecture (SOA) is the policies, practices and frameworks that enable application functionality to be provided and requested as sets of services published at a granularity relevant to the service Requestor, which are abstracted away from the implementation using a single, standards based form of interface.”

[21, p 6]

This definition has more focus on products, but not on technical products. Services are not the important issue here, it is the ability to deploy and control the Services.

The two examples I have chosen are non-technical. This is a deliberate choice as this aligns with my view on SOA. I could probably keep finding different definitions on SOA with various similarities and differences. In appendix A I have listed further 13 definitions, and looking at the two definitions I have quoted above, the problem is; I can’t take any of these 15 definitions and say they are wrong. I can point at issues that are questionable, but which can be explained from the perspective the definition is written. An example of such a definition is:

“SOA is a form of technology architecture that adheres to the principles of service orientation. When realized through the Web services technology platform, SOA establishes the potential to support and promote these principles throughout the business process and automation domains of an enterprise.”

Thomas Erl, chief architect, XML TC Consulting Inc

The definition by Thomas Erl is clearly a definition from a technical perspective, but is it wrong? I disagree with his view on SOA as a technical architecture, but as he calls Web Services a Technology Platform, which I see as a set of standards, the problem seems to lie in different interpretations of some of the underlying concepts, and not the definition of SOA itself.

⁹ Meaning where the actual systems are implemented, and by no means referring to level of knowledge.



My working definition of SOA will be that of CBDI, but with a note, that I see the CBDI definition as encompassing all the other definitions I listed in Appendix A. The A in SOA stand for architecture, and this is of course also a central aspect of SOA, but what really matters is the business design and delivery process [21, p. 4][37].

6.2.1 SOA and Web Services

Web Services (see section 6.2.4) has been the enabler of SOA [21, p. 4]. As Web Services often are used almost interchangeably with SOA, I will just spend a few words on this topic.

It is important to note that Web Services are not a mandatory part of SOA - SOA is a more general concept. Web Services are however the currently preferred method of practically implementing an SOA [21, p. 7]. Table 1 shows to some extend where Web Services fits in the SOA Picture.

Enabled by Web Services	Technology neutral	Endpoint platform independence
	Standardized	Standards based protocols
	Consumable	Enabling automated discovery and usage
Enabled by SOA (MAY be enabled by Components)	Reusable	Use of Service, not reuse copying of code/ implementation
	Abstracted	Service is abstracted away from the implementation
	Published	Precise, published specification functionality of service, not implementation
	Formal	Formal contract between endpoints, places obligations on provider and consumer
	Relevant	Functionality presented at a granularity recognized by the user as a meaningful service

Table 1: Web Services and SOA [Source: 21, p. 8]

The Web Service part in Table 1 shows how Web Services can provide the practical implementation of a Service, but that it is the SOA part that insures that the values of SOA are utilized.

6.2.2 The SOA Process

SOA should not be seen as a project that ends at some point. You want SOA to support the changes in the business, and the business is ever changing - hence SOA is ever changing. What you need to achieve is to make the ability to change as agile as possible.

The EA process is a known and described element of the EA discipline [5] [1], but the SOA-process is a newer concept with little supporting literature [13] [12]. I believe it is evident, that in order to realize the goals and promises of SOA, an ongoing process must be defined; The SOA-process. That SOA never ends, is a strong indication that SOA coheres with EA, and differentiates it from a normal development of software.

The following are bullets from SUN's¹⁰ "*Assessing Your SOA Readiness*":

- SOA is an architectural style that has been around for many years. While there are new ways to realize SOA, including the use of Web Services technologies, leveraging the experience of a services organization well-versed in SOA is essential to understanding technologies and techniques necessary to gaining the business benefits of SOA.
- Successful SOA is about more than deploying software. Organizations must evaluate their funding and governance models, analysis and design techniques, development

¹⁰ www.sun.com



methodology, deployment and support plans, and partner/customer/supplier relationships.

- Moving to SOA is no small feat. It can and should be done incrementally, but requires a shift in how we architect and compose services-based applications while maximizing existing technology investments.

The three main issues pointed out by SUN are the Technical, Non-Technical and the SOA Transition. These three issues support my earlier statement; SOA is not only a technical issue. Of course, everything that needs to be implemented has a technical side, but in order to utilize the technical possibilities it is the non-technical issues that can give the desired value.

In the following table I have listed some of the non-technical issues, which I see must be addressed in order to support and control SOA - e.g. issues that must be covered by the SOA-process.

Title	Implementation strategy - including a transition plan
Description	One of the big advantages of SOA is the possibility to make an incremental implementation and transition. However this requires a strategy on how to get from the "current state" to the "target state" [24].
Title	Evolution
Description	A SOA must be developed over time. The possible agility to gain from SOA does not come "in the package" but is a continuous effort.
Title	Organisation support
Description	Buy in from the organisation must be ensured. SOA affects the entire business. All processes in the business must be seen as Services in a SOA context.
Title	Monitoring Return On Investment (ROI)
Description	One of the SOA promises is to increase the ROI [22]. The actual benefits must be monitored in order to improve the weak points and learn from the good. Different parts of SOA can have a very different time-scope on ROI [12].
Title	Quality control
Description	Providing a Service for consumption means potentially providing for the entire business. This requires that all Services are subjected to continuous quality control.
Title	Service Oriented Development Method (SOAD)
Description	The introduction of SOA will change how development projects are executed [26].
Title	Concept definition
Description	SOA being a relative new concept must be defined for the business.
Title	Information architecture
Description	Communication in SOA is based on messages between loosely coupled Services. In order to insure a common understanding of the content of these messages all data ¹¹ must be defined in a common data model.

Table 2: SOA issues [Source: own work]

I will not describe the issues in further depth at this point, but return to these issues, and now look at how they relate to EA.

¹¹ Both semantic and syntactic definitions of all concepts in the organization



The above issues support the non-technical definition of SOA (see section 6.2), as all the issues are of a non-technical nature, but more interestingly the issues relates to what was identified as being in the context of EA (see section 6.1.1). I have mapped Scott A. Bernard definition of what is provided by EA; “Documentation method” and “Management program” (see section 6.1.1) into the context of Table 2.

SOA	Implementation strategy - including a transition plan
EA	- Current Architecture: Views of as-is strategies, processes, and resources - Future Architecture: Views of to-be strategies, processes, and resources - EA Management Plan: A plan to move from the current to the future EA
SOA	Evolution
EA	- EA Approach: A modelling framework and implementation methodology
SOA	Organisation support
EA	- Decision Support: Financial control and configuration management
SOA	Monitoring ROI
EA	- Decision Support: Financial control and configuration management
SOA	Quality control
EA	- Standardized Policy: Resource governance and implementation
SOA	Service Oriented Development Method (SOAD)
EA	- Standardized Policy: Resource governance and implementation
SOA	Concept definition
EA	- Resource Alignment: Resource planning and standards determination
SOA	Information architecture
EA	- Resource Alignment: Resource planning and standards determination

Table 3: EA and SOA relations [Source: own work]

EA is about identifying Artifacts and Components (EA-products), and describe and organize these in relation to one another. The issues related to SOA, that I have described in Table 2, will produce a lot of Artifacts and Components.

6.2.3 Services

The concept of developing Services in IT is not a new trend but has been used for ages, and is the part of SOA where the link to the established IT methodologies are strongest - as illustrated by hyperdictionary’s definition of Service:

“Work performed (or offered) by a server. This may mean simply serving simple requests for data to be sent or stored (as with file servers, gopher or http servers, e-mail servers, finger servers, SQL servers, etc.); or it may be more complex work, such as that of irc servers, print servers, X Windows servers, or process servers.”

[53]

This definition is not seen in a specific context, although it is centred around a server perspective.

CBDI defines a Service:

“...is a vehicle by which a consumer’s need or want is satisfied according to a negotiated contract (implied or explicit) which includes Service Agreement, Function Offered etc”.

[18, p. 5]

CBDI focuses much on the establishment of a contract, and not on where the work is being done. I like this definition although the definition of “Service” is somewhat fuzzy, but as almost all concepts it must be viewed in a context. The Service concept is a natural evolution on how we develop systems, and NOT a replacement! This is a very important point to remember as illustrated in Figure 6 the benefits comes from using it all together.

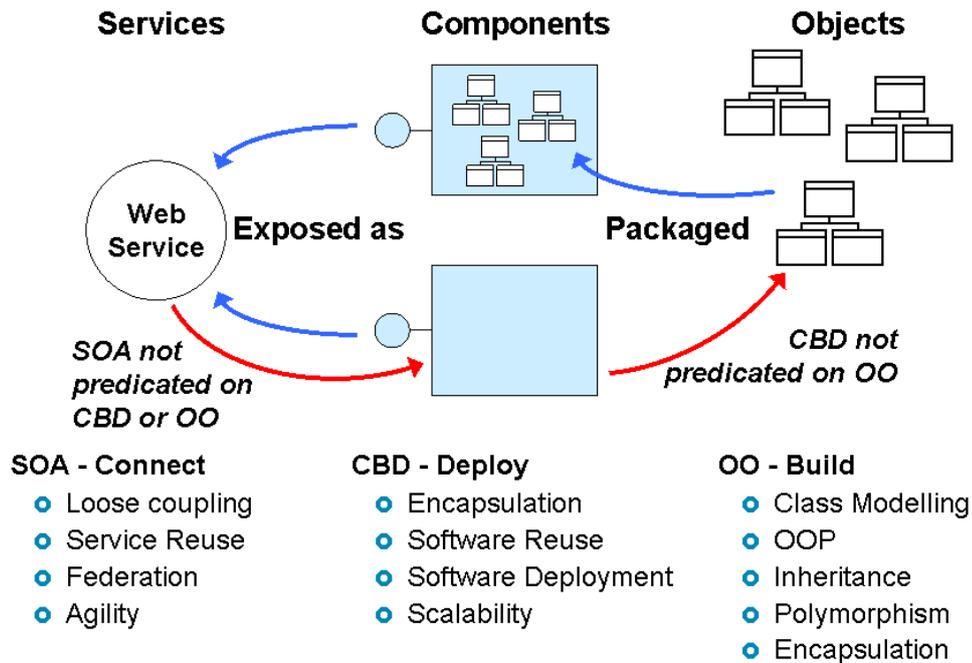


Figure 6: SOA, CBD and OO in Context [Source: 40]

Figure 6 identifies some of the differences between Services and Components - a discussion I have often been witness to. It is important to note that they can support each other, but that they focused on different aspects.

That being said SOA isn't necessarily build on component technologies, and similarly Component Based Development (CBD) isn't necessarily build on OO [40]. However I will not go further into the discussion on Services vs. Components, but move on to looking at how the Service concept can be practically implemented in an SOA.

However before I continue I will leave you with this little quote by Mark Twain:

"History does not repeat itself, It rhymes."

[76]

What it illustrates in this context is that, as was the case with CBD, SOA based on a lot of the experiences, and will on many issues be similar to it "legacy".

6.2.4 Web Services

When in the field of SOA the concept of Web Services is unavoidable. It must be emphasized that a Service is not equal to a Web Service. A Web Service is a practical implementation and is currently the de facto standard of implementing Services in a SOA. As with SOA, Web Services have also been through the process of "hying".

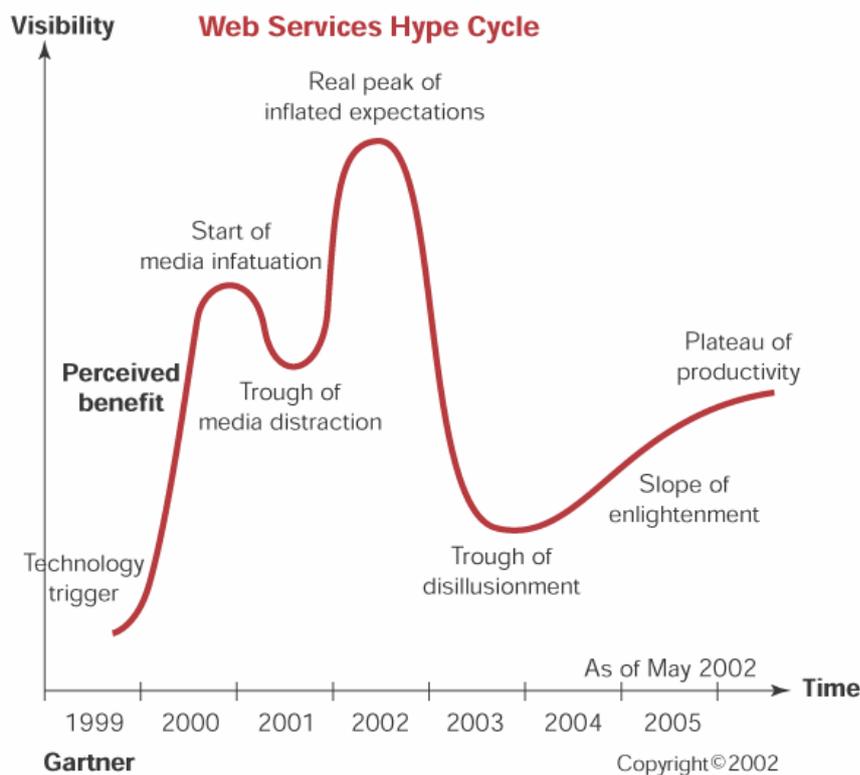


Figure 7: Gartner Web Service Hype Cycle [Source: 130]

The figure originates from 2002 so it is interesting to see how Web Services fit in the Gartner Hype Cycle today. One can argue about the correctness of the level of visibility but the different phases described in the figure seem to have almost hit spot on.

Where the concept “Web Service” differs from the previous definition¹², is that this is meant for physical implementation. Although this is the case, one can also look conceptually at Web Services. I have had several discussions with people about what a Web Service is. One in particular states very clear that the definition has changed over time. The discussion originates from a discussion I had with a professor¹³. His definition of a Web Service was; all systems providing dynamic content on the web (e.g. cgi, asp, servlets etc.). My definition was much stricter. So as an attempt to clarify the issue I asked the author of “Web Services Essentials” by Ethan Cerami, who returned with the following:

“First off, many people define web services differently. Some people define it as any web site that provides a service. For example, the Google web site provides a web search service. This is probably how your prof is defining it. However, today, when most people say “web service”, they usually no longer mean this. My definition of a web service is any service that is “application-centric”. By this, I mean, it’s:

1. available over a network.
2. uses a standardized XML messaging system.
3. not tied to any one OS or platform.
4. self-describing (although not required)
5. discoverable (although not required)

¹² As Web Service is a practical view a service.

¹³ Teaching; Interactive Web Services with Java and XML



The important piece is XML, as XML enables other computer applications to interact with it. A web service that just returns HTML (e.g. cgi, asp, etc.) is primarily created for human consumption, and I wouldn't include it in the definition of "web services".

[Ethan Cerami, contacted by mail]

The important thing to note here is that he points out that the definition of a Web Service has changed over time. What is also interesting is the mix of conceptual and more specific demands that Web Service must comply with - this is where we hit technology. As Ethan Cerami also points out; Web Services are intended for machine-to-machine communication.

Currently the w3c definition is:

"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."

[48]

The reason I bring the definition of w3c to attention is because it too have been subject to change. The original definition was:

"A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols."

[121]

My own definition has also changed since I started looking at the subject. I have also been implementing Web Services and experienced the problems that can still arise, even if complying with the above definition. This has resulted in my practical definition:

A Web Service must:

1. comply with the w3c definition
2. comply with the definition by Ethan Cerami
3. comply with WS-I¹⁴
4. be loosely coupled

I call this a "practical definition" because it is essentially what it is. I am not arguing the definition of w3c, I am only extending it in order to make it applicable within a SOA context.

6.2.5 Loosely coupled systems

That Services in an SOA are loosely coupled means that the interactions between the Services are not hardcoded. The connection will be effectuated at runtime making it possible to achieve independency of the Service location - which is also one of the sales pitches listed in section 5.

SOA can essentially through the use of standards be viewed as the glue between loosely coupled systems, but as with SOA, Loosely Coupled Systems (LCS) can be defined at many levels of abstraction. To illustrate this I have listed definitions from three different sources:

¹⁴ WS-I is an open industry organization chartered to promote Web services interoperability across platforms, operating systems and programming languages. The organization's diverse community of Web Services leaders helps customers to develop interoperable Web Services by providing guidance, recommended practices and supporting resources. All companies interested in promoting Web Services interoperability are encouraged to join the effort. [source <http://ws-i.org>]



“A good working definition: loosely coupled is an attribute of systems, referring to an approach to designing interfaces across modules to reduce the interdependencies across modules or components - in particular, reducing the risk that changes within one module will create unanticipated changes within other modules.”

[49]

“loose coupling - The friction-free linking enabled by web services (or any SOA). Loosely coupled services, even if they use incompatible system technologies, can be joined together on demand to create composite services, or disassembled just as easily into their functional components. Participants must establish a shared semantic framework to ensure messages retain a consistent meaning across participating services.”

[50]

“Coupling is the dependency between interacting systems. Dependency can be classified as real dependency and artificial dependency. Real dependency is the features or services one consumes from other systems. Artificial dependency is the thing one has to follow in order to consume the features or services provided by other systems. Typical artificial dependency in IT are: language dependency, platform dependency, API dependency ...

There are two rules here:

- *One can never reduce real dependency but itself is evolving.*
- *One can never get rid of artificial dependency but one can reduce artificial dependency or the cost of artificial dependency.*

Hence, loose coupling describes the state when artificial dependency or the cost of artificial dependency has been reduced to the minimum.”

[51], [52]

If one does not look at the different levels of abstraction the three definitions agrees on that a LCS must comply with:

1. must be defined from its interface
2. must communicate using messages
3. all parties in a network of LCS shall communicate using a commonly defined data model
4. must be able to communicate across different platforms - being platform independent.

Theoretical definitions, like Dr. Hao He points out, have a practical approach. As he points out, a system never will be completely loosely coupled, but that a LCS is a system with as much coupling removed as possible.

Doug Kaye has also looked at this issue. In the following illustration he focuses on the differences between Loosely- and Closely coupled systems.

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

Figure 8: Loosely- vs. Closely coupled systems [Source: 8, s. 133]

The differences between tightly- and loosely coupled in Figure 8 the gives a good indication of the extend of change SOA will bring. The particular interesting row is the “Consequences”. What he is referring to, is when you make a LCS and opens up for use by the defined interface you don’t know the future use scenario [8, s. 144]. The consequence being you might be facing scenarios that the Service can’t handle - but it is also an indication of success as the Service is used!

Loose coupling is often seen as the road to agility, but this line of thought is often followed by the misconception that loose coupling is only a technical issue. I have in this section discussed loose coupling as being a technical issue, but I only see this a part of the picture loose coupling is multi layered. The following layers are based on the work of CBDI [40]:

- Business
- Organization
- Information
- Semantics
- Process
- Device

I will not go into further depth with the discussion of loose coupling, but emphasize that the concept can, as the concept of SOA, be used in many different levels of abstraction.

6.2.6 Web Service Levels

Often the actual Web Services¹⁵ are divided in two categories; Primitive- and Complex Web Services. As with almost all other concepts in SOA there are divergent perceptions and even different words for the same concept.

Primitive Web Services can be viewed as the *base* Services of the organisation. David Sprott describes these as “entity components”. These should provide the simplest building blocks of SOA. The primitive Services can use other Services, it is seen from a business process perspective they must not depend on other business processes. The primitive Web Services are not a business process by themselves, but common functionality that is used across different domains in the organisation.

¹⁵ I will from this point use the terms *Service* and *Web Service* interchangeably.

Complex Web Services are the mapping of the business processes into Services - into the ratio; one process one Service. A Complex will usually make use of other Services which can be both Primitive and Complex. David Sprott has named the concept “process components” and is furthermore advocating for these should be build in compliance with the “Mediator Pattern”¹⁶ [42][20, p 11].

Further Classification of Web Services, for both Primitive and Complex Web Services, will often be done. The reason for this can be issues as Service federation, improved overview of Services, and many others [20, p11].

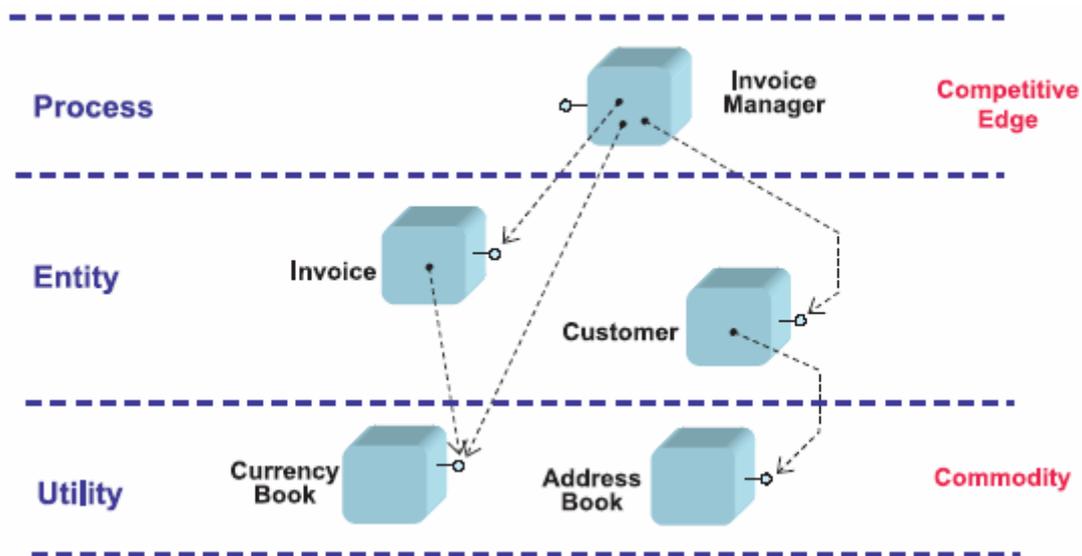


Figure 9: Service Levels [Source: 20, p11]

From an architectural viewpoint the different levels provides a good tool for defining different requirements, such as use of standards. The characteristics of the levels on Figure 9 are defined as follows [20, p11]:

Process Services are the actual business processes emulated in Services, and hence are likely to be built in-house, or be based on a package that allows extensive customization.

Entity Services have very few business rules attached to them other than basic validation through to high-level processes. Entities shall be “entities” in themselves, irrespective of which process that invokes them. An entity may make use of bottom-level “utility” components.

Utility Services can be viewed as commodity Services. Examples of Utility Services are data access components.

On Figure 9 “Competitive Edge” and “Commodity” are placed to emphasize that “Process Services” are where the business should gain increased competitiveness by creating a flexible system of Services. That “Utility Services” should be seen as a commodity refers to the simplicity of this level of Service.

Again it must be emphasized that there is no general best practise on this area - other than dividing Services in some form of grouping is a good idea.

¹⁶ The mediator pattern is a very general pattern with the purpose of coordinating the group of actors in a way such that the participating actors don't need knowledge of each other [6, p 509]

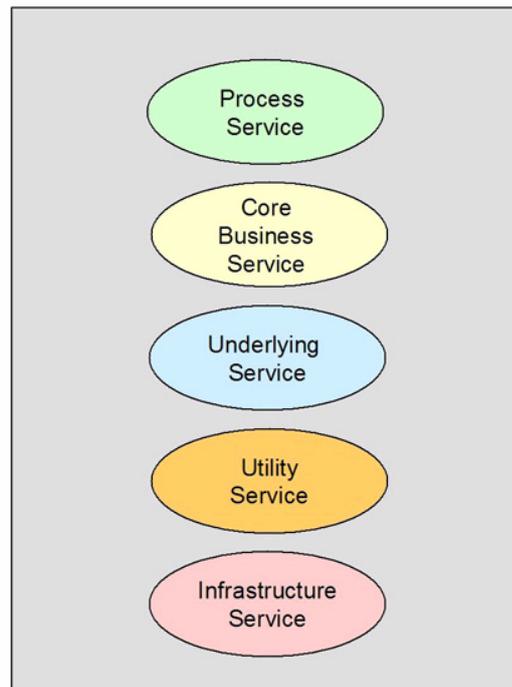


Figure 10: Service ontology [Source : 46]

Figure 10 divides the Services in five categories but David Sprott and Lawrence Wilkes agrees that there is no general practice on this area and organizations may chose to omit or ad levels to the Service Layers [46, p. 4].



6.3 Summary

Throughout this chapter I have looked at some of the most central concepts of SOA. This included not only the concept of SOA but also related concepts, both directly derived from SOA but also concepts of classic computer science. This was done as a consequence of the much differentiated views on SOA related concepts. The purpose of the chapter was not to explain the concepts in depth, but give my view of the concepts in order to avoid misunderstandings based on different understanding of concepts - or in other words; set the context for the further work of this thesis.

In this chapter I have seen EA as related to SOA, as I in my thesis problem have chosen SOA as my reference point. Many would probably argue that SOA that is related to EA and not the other way around. As this issue is a very central of part of this thesis I promise that this issue will be discussed in depth in the following chapters.

Although I have covered several concepts of the SOA world I have had to leave out many other interesting concepts, such as:

- **Service oriented Analysis and Design (SOAD)**
A method not yet developed, though discussed by many. Is it possible to combine all the disciplines of enterprise software development in one? Or is the goal just to adapt methods like RUP¹⁷ into SOA-RUP?
- **Granularity**
I have in this section discussed the concept of a Service, but when actually implementing these Services the question of granularity becomes imminent. How much functionality should the individual Service encompass? I have had discussions with several companies¹⁸ on this issue and they have all seen this as a major challenge.
- **Service Catalogue**
The UDDI¹⁹ is often seen as a Service Catalogue. I will however postulate that the UDDI is not fit for the task of organising thousands of Services. It will not be possible to identify if a Service that fits your needs is registered in the UDDI.
This issue is one of my favourite questions to ask enterprises. I have asked Nykredit, Danske Bank, A.P. Moller - Maersk Group and several others, and all have answered that they know the problem will come, but that the haven't got a solution. Let us not repeat the flaws of Component Based Development (CDB) where we have big component libraries that no one really could use.

Several other issues could be noted here, but as initially stated it is not my purpose to explain all the concepts around SOA. The purpose of this chapter was to clarify how I see SOA to ensure that the following chapters are seen in the same view as they were written.

¹⁷ Rational Unified Process

¹⁸ Such as Danske Bank and A.P. Moller - Maersk Group.

¹⁹ Universal Description, Discovery, and Integration (UDDI): Is a standard for a platform-independent, open framework for describing services on the Internet.



7 Evolution and maturity of SOA and EA

The focus of this chapter will be to look at how SOA fits within the big picture of IT and Software Development, which means looking at the origin of SOA, and the evolution in the IT world in general. Sometimes one might think that the new concepts are a part of a grand marketing scheme of the consultancy world, in order to hype concepts making it possible for them to keep themselves busy. SOA has been subject to a great amount of hype, but I also see that SOA has evolved dramatically throughout the period of hype. I will take a historical view upon SOA in order to identify the general evolution of SOA, and will also use this section to clarify my own view on SOA.

SOA is not something you just get over night. My last subject of this chapter is to have a look at how the current state of SOA can exist in different stages. In the world of EA this issue is referred to as maturity and I will use the experiences from this area and use these to identify the connection between EA and SOA.

7.1 SOA and software development

The Top-down approach of EA vs. the Bottom-up approach of SOA has a big part of dilemma when looking at EA and SOA [23]. The dilemma being; that SOA might originate as a bottom-up paradigm, but SOA has evolved to cover a much broader spectrum, and should no longer be seen a technological Bottom-up project.

Bottom-Up Approach Point Projects	Top-Down Approach Area Projects
Local short-term initiative	Broader, longer-term initiative
Building a solution against immediate requirements (where “building” means design, construct or assemble)	Focus on system properties across a whole area (e.g. business domain, technical domain, infrastructure)
Strongly aligned to local objectives.	Creating value by establishing (procuring or building) conveniently available resources
Cost-effective use of conveniently available resources (improvisation or “bricolage”)	Indirect links between benefits (across area), costs and risks
Direct link between (local) benefits, costs and risks.	<ul style="list-style-type: none"> • Often difficult to create/maintain business case for adequate investment in resources and infrastructure
No mandate to pay attention to broader, longer-term opportunities and effects.	<ul style="list-style-type: none"> • Often difficult to demonstrate return on investment

Table 4 : Bottom-Up versus Top-Down [Source 109]

Table 4 points out some general effects of approaching projects Bottom-up vs. Top-down. As SOA is an enterprise-wide endeavour the most important issue is to have the benefits of approaching Top-down in order to have the broad view of the program.

Essentially working with IT has always been about abstraction and conceptualizing your business into being IT-supported, and looking at the history behind EA and SOA I clear indications of the two worlds merging - and why. EA is, as described in section 6.1.1, founded in the world of business, whereas SOA still have strong ties to the world of software development. As I pointed out in section 6.2.3 SOA is often seen as Component Based Development (CBD) - “putting old wine into new bottles”. This is however not how I see it, and as illustrated on Figure 11 (repeated from section 6.2.3) SOA is not a replacement of CBD.

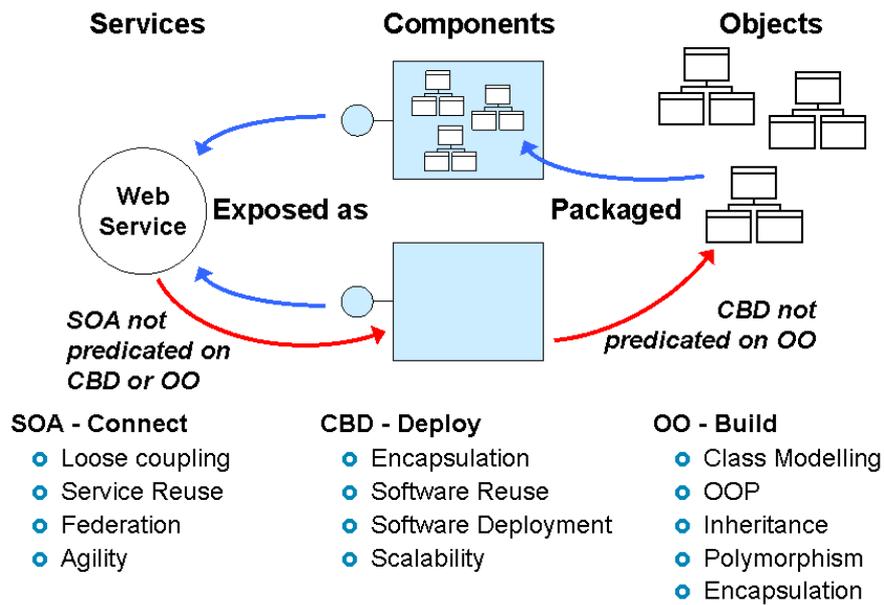


Figure 11: SOA, CBD and OO in Context [source: 40]

As the Figure 11 show Services does not replaces the paradigms of CBD and OO, but they will coexist. SOA is a new level of abstraction on software engineering, as was CBD on OO [40, p. 22]. And as stated, heightening the level of abstraction has always been an issue in Software Development [56].

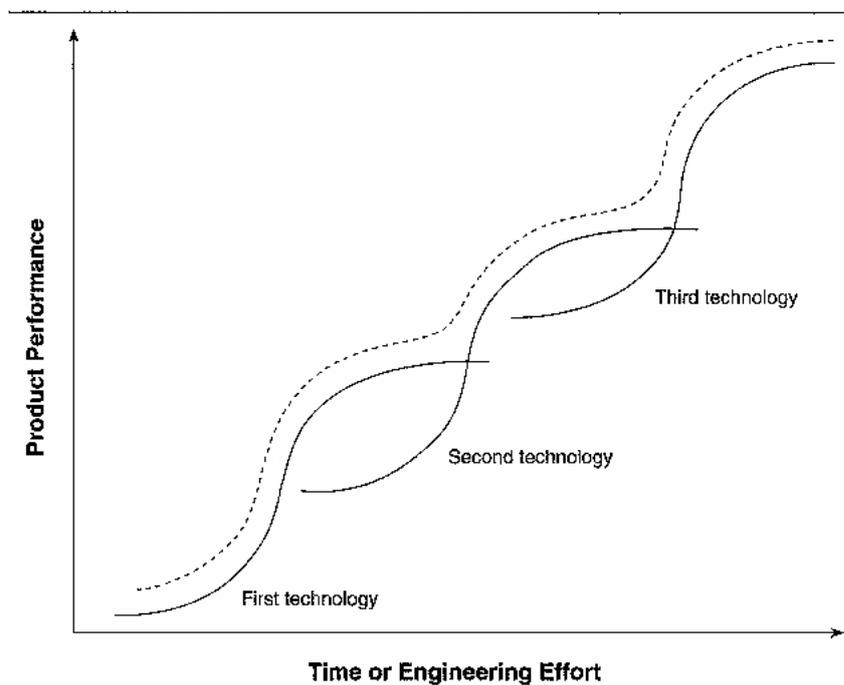


Figure 12: The Conventional Technology S-curve [Source: 58 p. 40]

Jack Greenfield makes a reference to Clayton M. Christensen’s “*Conventional Technology S-curve*”, shown in Figure 12 with a reference to the shift in paradigms that have been seen with regards to “the popular” software paradigms throughout time[56]. This is exactly how I see SOA, in the context of software paradigms; it is a step in the natural evolution.

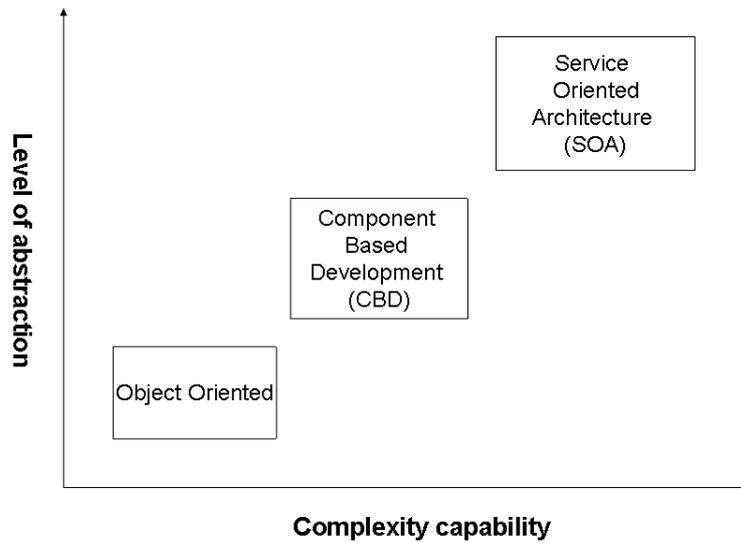


Figure 13: Evolution of paradigms [Source: own work]

Figure 13 is an illustration of the evolution from OO to SOA. The x-axis is the “complexity capability”, which is the ability to encompass the demand to organise an increasing complexity in a human comprehensible fashion. The y-axis is the means, in terms of increasing the level of abstraction. These characteristics have been predominating throughout the evolution of software paradigms [56], and SOA is “just” another step on the evolutionary ladder.

7.2 Evolution is constant

The only constant is change. Geoffrey A. Moore has with his “*Technology Adoption Life Cycle*” approached this very issue, and developed the following model:

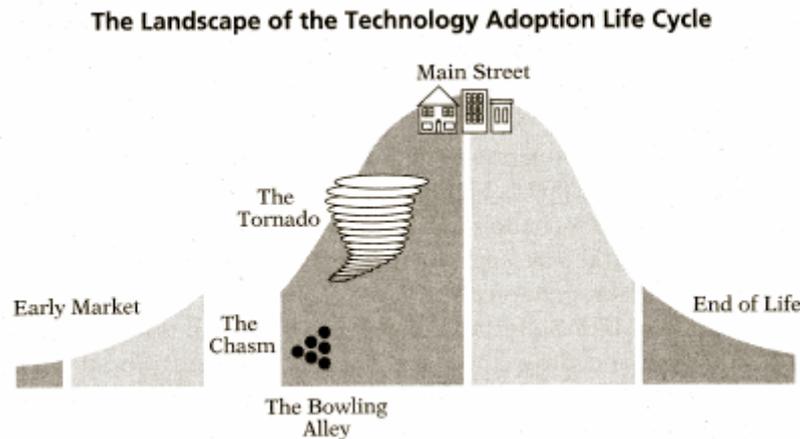


Figure 14 : The Landscape of the Adoption Life Cycle [Source: 75 p. 25]

The table below contains a short description of the different phases:

Early Market: a time of great excitement when customers are technology enthusiasts and visionaries looking to be first to get on board with the new paradigm.
Chasm: a time of great despair, when the early-market’s interest wanes but the mainstream market is still not comfortable with the immaturity of the solutions available.
Bowling Alley: a period of niche-based adoption in advance of the general marketplace, driven by compelling customer needs and the willingness of vendors to craft niche-specific whole products.



Tornado: a period of mass-market adoption, when the general marketplace switches over to the new infrastructure paradigm.
Main Street: a period of after-market development, when the base infrastructure has been deployed and the goal becomes to flesh out its potential.
Assimilation: the technology loses its discrete identity, moves into decline and is supplanted by a new technology paradigm.

Table 5 : The phases of the Adoption Life Cycle [Source: 78]

The model originates from Geoffrey A. Moore’s book of 1991”*Crossing the Chasm*”. The description below is a short elaboration on the subject:

“Moore’s theory is built on the idea that the rate of diffusion in the Technology Adoption Life Cycle curve is not continuous in high tech markets. Moore borrowed his diffusion of innovations theory from Everett Rogers²⁰, but argued that there is a chasm between the early adopters of the product (the technology enthusiasts and visionaries) and the early majority (the pragmatists). This is because visionaries and pragmatists have very different expectations. Moore exposes those differences and builds from there to suggest techniques to successfully cross the chasm, including choosing a target market, understanding the whole product concept, positioning the product, building a marketing strategy, choosing the most appropriate distribution channel and pricing.”

[76, Crossing the Chasm]

The key element is “The Chasm”, but also that this is not a “hype cycle” as Gartner sees it! Moore’s view on the model is based on marketing issues of technology, but what he also points out is that the marketing abilities changes with the evolution of the technology. When the technology matures it gets easier to adopt, and ones expectations will be more realistic. Nicholas G. Carr wrote the article “*IT Doesn’t Matter*” [77], which is another way to put the point’s made by More. It is not the technology that will improve your business, it is how and when you use the technology available. So, the key question is if the technology, supporting the ideas of SOA, have the right level of maturity making SOA the right choice in time. I will return to this in the next section, but for now look at bit further on SOA.

Moore’s “*Adoption Life Cycle*” is, as the title insinuates, focused on technology. It is however my claim that the model is much more general. The “*High-Tech Sector Growth Model*” (see Figure 15) Moore introduces is a complete parallel to Clayton M. Christensen’s “Conventional Technology S-curve” (see Figure 12).

²⁰ http://en.wikipedia.org/wiki/Everett_Rogers

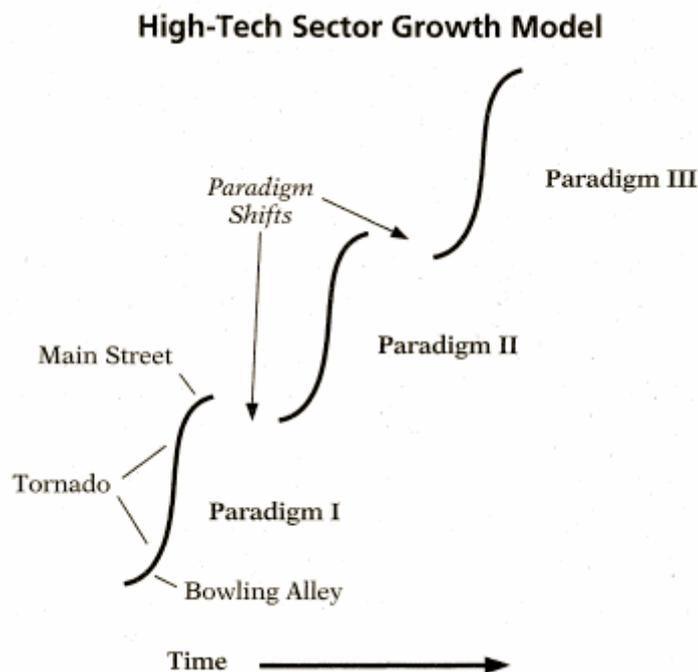


Figure 15 : High-Tech Sector Growth Model [Source: 75. p. 105]

In the High-Tech Sector Growth Model illustrated in Figure 15 Moore is referring to shifting in paradigms. SOA is also a paradigm, and as I discussed in section 7, SOA is “just” another step on the evolutionary ladder making SOA a perfect fit into High-Tech Sector Growth Model.

If EA is also a paradigm, then following the High-Tech Sector Growth Model EA will also be “just” another step on the evolutionary ladder. If this is the case, both EA and SOA will be replaced over time - by what can only be a result of speculations upon the future. It is in my view however evident that both EA and SOA will change over time, as they have in the past - change as a result of the maturity process (see section 0).

Accepting that I can’t predict the future I will limit my view to a single Life Cycle. The maturity process is parallel to the Life Cycle concept, as the maturity process will only exist within one Life Cycle. When a concept or a technology is at the end of its Life Cycle, it will be replaced by a new concept or technology, starting a new Life Cycle and maturity process. This often builds on the previous experiences, but will be self contained.

As stated in section 6.1.1.1: The driver for EA is essentially “The Red Queen Principle” [69]: What you want from your EA is the ability to “run” without straining yourself. Often the general notion is that EA should be constant over time, regardless of the technological and business changes [71][72, p. 30]. I see this view as being unrealistic, EA is not supposed to be constant [67][68]. The desire should be, as in all other levels of the IT world, to create a system or framework that can encompass change. But to anticipate all coming changes is not possible! It might be possible to have an EA that is constant over time that can keep your company “running”, but if you don’t adapt it to the changes around you, you will not be able to “run” fast enough to even stay at the same place.

“In today’s world, change is the only constant, and the ability to manage that change, is the only competitive advantage.”

[73]



To view technology and economics as being independent of one another is in my view impossible. A clear example of this, is the influence the Internet has had on how businesses work and cooperate. The boundaries between the companies are changing, approaching the “connected economy” [67]. An EA should not be viewed as being static. It should, as almost everything else in the IT world, be created to encompass as much change as possible.

“An architecture, enterprise or not, must evolve over time to reflect the new business needs and technological approaches. When an architecture is outdated, or perceived to be so, the chances of project teams adopting the architecture decrease dramatically. Architecture models are not carved in stone – they are living things that are updated on a regular basis.”

[67]

The statement above is where I see SOA having a very big impact on how the business should conceive IT - as an example in relation to the Business Process Modelling (BPM), the business processes are Services. If you have an existing EA it will be necessary to re-evaluate if it can encompass the challenges of SOA in a way making it possible to harvest the advantages of SOA. Another possibility is, if the organisation is beginning to implement an EA from scratch based on SOA, which is often the case [62]. This also means that depending on the maturity of organisation, SOA will have different implications.

7.3 The evolutionary phases of SOA

In section 6.2 I discussed the definition of SOA, which proved to be a choice of definition amongst many possible candidates. One of the reasons of the diverse definitions, is that SOA crosses over many levels of IT, but there is also another reason for the diversity of SOA definitions; SOA has gone through an evolution and has matured over time.

Jeff Schneider has made an interesting post on this subject [79], which builds on the notion that SOA has gone through three phases. Below I have quoted Jeff Schneider for his three phases - written like a true practitioner:

- **Phase I:** was basically 'protocol oriented' - with a focus on the WS-I Basic Profile (SOAP, WSDL, UDDI, XML Schema).
- **Phase II:** was the WS-* stack (the aspect oriented protocols).
- **Phase III:** is about the using the darn things. Whacky, eh? You see, I talk with hundreds of companies about their utilization of Web Services. In most cases they tell me this:
 1. We connect fat .Net clients to Java application servers (and use soap in between)
 2. We connect to some ASP²¹ (like Amazon, SF.com, etc.)
 3. We front ended some legacy system with Services.

It's my belief that Jeff Schneider is on to something very essential here, however there is a pre-phase to what he calls phase I, but first I will look closer at the phases I to III:

SOA Phase I

As Jeff Schneider points out this phase was protocol oriented. The reason for this focus was, that even though standards were defined in order to ensure interoperability, there were some practical problems. Standards are usually written to cover a very broad spectrum of scenarios which means they need to be very general. Unfortunately this means that they often will be open to interpretation [81][82].

Another issue is that the standards, in order to accomplish the generality they, often leave open “slots” which are to be defined in the given implementation [80]. The standard will define the legal input in these slots, hence ensuring the interoperability through

²¹ Application Service Provider



documentation. However, this approach will often not stand the test in the practical world. An example of this could be the following case where an open “slot” is to identify which encryption algorithm is used:

SSL²²/TLS²³ can use RSA²⁴, DSA²⁵, or various Diffie-Helman algorithms to exchange an encryption key. The actual encryption options include DES²⁶, AES²⁷, RC4²⁸, and so on. With no specific configuration, for example, openssl supports more than 25 combinations, which doesn't count some parameters such as the RSA key size (512, 1024, and 2048 bits are the most common) -- we really have over a hundred possibilities.

With so many options, ensuring interoperability across platforms can be difficult, if not impossible. In SSL/TLS, two vendors could each implement a couple dozen choices and still have nothing in common. Or a group of interested participants could get together and profile SSL/TLS, resulting in an "SSL profile" that specifies a common subset that everyone will support.

[80]

It is problems like these WS-I solve by specifying an interpretation of the standards, and identifying a subset of what is applicable into the “slots”. So WS-I is not a new standard, but a specification of how the standards are to be used in practise.

This is the 'protocol oriented' aspect of SOA Phase I; it was found that in the practical world of implementing Web Services didn't ensure the desired interoperability across platforms. These problems were solved with the WS-I initiative and we can move on to the next phase²⁹.

SOA Phase II

Where the WS-I Basic Profile ensured interoperability of the “simple” parts of messaging through Web Services, the next phase was to support the surrounding aspects of messaging; such as security, reliability, transactions, addressing etc (see Appendix B).

Essentially the issues covered by the WS-* stack are common problems in the world of computer science. Hence the WS-* stack is not a solution to unsolved problems, but the effort to standardize how to approach these issues.

SOA Phase III

Jeff Schneider's view here is that this is the point where it makes sense to talk about real interoperability, and that this is where real implementation can begin. So from a SOA viewpoint this is essentially not until this phase that SOA will be realistic.

SOA Pre-Phase

In my view a Pre-Phase or Phase 0 also exists, where the work on creating the Web Service concept was done, and where the initial implementations was done. It was from these experiences that the need for what Jeff Schneider is pointing out arose (as discussed above).

²² Secure sockets layer.

²³ Transport Layer Security.

²⁴ The letters RSA are the initials of developers surnames: Ron Rivest, Adi Shamir and Len Adleman.

²⁵ Digital Signature Algorithm.

²⁶ Data Encryption Standard.

²⁷ Advanced Encryption Standard.

²⁸ ARCFOUR.

²⁹ It should be noted that it is only the basic messaging that is covered by WS-I basic profile.

7.3.1 SOA Phase IV

Jeff Schneider’s Phases is about the evolution, but also about an increasing maturity of the concept of SOA. Other people have also started to look at the concept of SOA maturity [83][84][85][86][87]. Comparing these views on SOA maturity is however difficult as they are defined on different levels of abstraction, and even with different comprehensions of the concept SOA! However there are two things in common for all the views on maturity:

- The fact that SOA has, and is maturing,
- and that the maturing process can be divided in several phases or levels.

7.3.2 The EA phase of SOA

The phases of SOA are also an interesting point that can help explain the many definitions of SOA (See appendix A). SOA has changed over the course of time, and the SOA Phases should be seen in the light of the definitions. Initially SOA was a very technical founded, and it is the evolution of the Web Service standards and the SOA Phases from I - III that have enabled the realization of SOA.

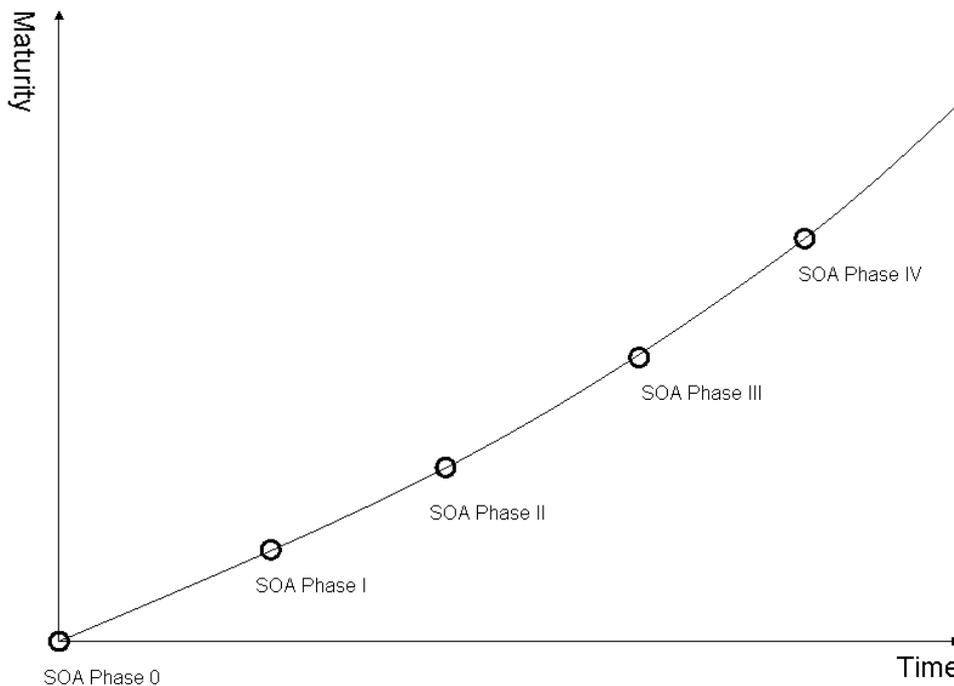


Figure 16 : SOA Phases [Source: own work]

On Figure 16 I have illustrated the dependency between time and maturity. The figure is not supposed to stand alone but should illustrate the importance of including the maturity aspect when discussing SOA. SOA is not just SOA, it will be a product of where one has reached in maturity.

When we enter SOA Phase III we are “crossing the chasm”, at least from a technical perspective. But in order to really get the value from SOA we have to move on to the next phase; “SOA Phase IV”. As stated previously it is after the technical hurdles (after phase III) where SOA gets interesting, and is what my thesis problem is all about. When SOA is implemented as a physical system, the challenge of keeping both the agility and control over time will be the predominant concern, while keeping the IT and business tied together. At this point SOA’s resemblance to EA is getting more and more visible. The common goals of EA and SOA, such as creating interoperability, reuse and eliminate duplication can to some level be reached running a “clean” SOA-project. But the true value lies in combining EA and SOA -



together they can guide interoperability, not only between IT and IT, but between the business and IT - Enterprise-wide interoperability [23].

7.4 Parallels of SOA and EA

Looking at EA and SOA, it is exactly the evolution in software engineering, as depicted in Figure 13 that illustrates the shifts to more abstract paradigms that makes SOA different and not “just” another step on the evolutionary ladder. EA is currently the most abstract view of IT, and SOA is rising into the same level of abstraction, equating the top-down view on IT as EA. I am not equating EA and SOA but there is a common level of abstraction on which they both operate. However they furthermore cover very different levels of abstraction.

There are those who equate SOA to EA [33][36][70]. But SOA is not EA, and EA is not SOA; the similarity lies in the shared goals [23], and in order to get an enterprise-wide value of SOA, it is imperative that EA and SOA are thought of as supporting each other.

"If you don't do EA, you can't do SOA." was stated by Gurpreet S. Pall on his presentation on the TechEd in Malaysia 2004 [74]. This is a somewhat challenging statement in which I both agree and disagree. My understanding of Gurpreet S. Pall's statement is; If you look at the definition of SOA, it is possible to do SOA without EA, but in order to use SOA to its full potential EA and SOA must be seen as interrelated.

EA and SOA can exist in many forms, which is a result of one of the recommendations often made to organizations when starting on EA and SOA:

- EA is not just EA, but should be adapted to the needs of the individual organization,
- and SOA is not just SOA, but should be adapted to the needs of the individual organization.

In order to combine SOA and EA, it is my belief that both must be at a similar level of maturity. This is of course not possible from day one, but aligning the maturity levels should be the vision of where one wants to bring EA and SOA - it would make little sense to look at how SOA in phase I (as discussed in section 7.3) will affect a fully matured EA.

One of my goals of this thesis is to identify the relations between EA and SOA. I will use Peter Herzum's article "Applying Enterprise Architecture" [57] as this article focus on the maturity of EA, making it possible through his views on EA, to map where SOA interrelates with EA. The purpose of this "exercise" is to state how I see SOA in relation to EA and EA maturity. But before proceeding to look at maturity I will, based on Herzum's article, look at some of the more general issues of EA and their relation to SOA.

I will discuss the issues of; SOA is enterprise-wide, Architecture analogy and Green field³⁰ projects in the following:

SOA is enterprise-wide

The E in EA stands for "Enterprise" but it could also mean "enterprise-wide". It covers all aspects of IT, including user interfaces, portals, new and legacy applications, technical infrastructures, and databases. [57, p. 3]

SOA is not just a project resulting in a system that can be added to ones existing system portfolio - SOA is the system portfolio! This is in my opinion there where SOA is fundamentally different from existing software development. The idea of having one system to cover the whole organisation is however not new. Enterprise Resource Planning³¹ (ERP)

³⁰ Green Field is referencing to starting from scratch - meaning that you won't be limited by existing applications.

³¹ See Appendix D



systems were the popular choice (or hype) throughout the 1990's [88]. The purpose of these systems was similar to SOA; to package everything into one system, covering all the needs for the entire enterprise. However the lessons learned from these ERP projects resulted in having numerous unintegrated systems running:

"...a large enterprise typically operates five or more ERP systems and some companies are known to have more than 20 ERP systems "

[90].

One of the major drawbacks of ERP systems is that they are too inflexible as they are implemented from a given scenario in time. Going SOA will give an enterprise-wide view of the processes of the enterprise - not only as a piece of software. It should be noted that ERP-systems can be a part of a SOA.

Just to emphasize that SOA will not be the "silver bullet" I will bring a quote from Jeff Schneider's Blog: Service Oriented Enterprise.

I was chatting with a CIO the other day and asked him, "In retrospect, do you think you were involved enough in the early stages of your ERP implementation?" He answered, "In retrospect - no - I wasn't involved enough. Had I known the size of it I would have definitely gotten more involved."

I followed up with, "Are you aware that an enterprise SOA roll-out will be significantly larger than your ERP implementation?"

He started laughing; he thought I was joking. My face didn't change. He quit laughing. "Jeff, are you serious?"

"Yes, I'm very serious. SOA is a complete overhaul impacting how systems are analyzed, designed, built, integrated and managed. And not just some systems - all systems including packaged applications like ERP."

[96]

I believe that this is a good illustration of how big an endeavour SOA is, and that the consequences of not taking SOA seriously can be catastrophic. This is focused on the implementation, but there is also another issue to the realization of "enterprise-wide". Over time, it is necessary to address the whole software lifecycle, including maintenance, evolution, retirement, distribution, and the operation of applications [57, p. 3]. This is perhaps the most complex aspect of both EA and SOA, but my vision is; that by not looking at EA and SOA as to separate disciplines, the complexity can be reduced.

Architecture analogy

In order to grasp complexity it is always a good idea to try with an analogy to a similar concept that can eliminate some of the complexity of the actual concept one is trying to understand. Herzum is also using this method when trying to illustrate EA, ending up by viewing EA as an organic³² system. This is a complete match on how I see SOA, in fact so much that my original title for this thesis was "SOA - the organic Information System (IS)".

The building analogy is too limiting for EA as it doesn't recognize the dynamic nature of EA. This is also recognized by mature enterprises who will see EA analogous to urbanism and even ecology, the goal is; by only defining principals, to allow a set of independent systems to evolve over time while still achieving the overall vision [57]. The analogy illustrates the scenario but not how it is achieved in an EA context. This is exactly where SOA enters the picture - SOA is all about agility to support the dynamic nature of enterprises.

EA is by some viewed as a "system of systems", hence the reference to ecology, but as Herzum points out, in the case of EA, this approach does not scale. Originally this was Zachman's motivation (see section 6.1.1.1), but as previously discussed both EA and SOA

³² Herzum uses the terms urbanism and ecology.



has evolved.

What's interesting is that SOA for sure can be viewed as a "system of systems", and the quote of Aristotle; "The whole is greater than the sum of its parts" makes it clear, SOA should not be viewed as a collection of systems! SOA is a paradigm that elevates the abstraction level, which eliminates the focus from the actual systems to the interfaces, and secondly should provide support to view the sum of its parts as a whole - which is where the real value lies.

Why do I see SOA as being organic in nature? If SOA removes the focus from systems using interfaces, then it is no longer a system of system. This is exactly my point; I believe it is possible to control "the beast", but in order to do so, it is necessary not only to define the principals of the organism but also having a way of controlling that the principals are followed. And just as important the principals aren't constant; there is a need for mechanisms that support change and evolution.

Green Field projects

Imagining an Enterprise with no IT legacy would be unrealistic, or in other words; there are no Green Field projects left. The previous Architecture analogy miss the fact that EA and SOA projects will not only be about setting the rules for the new architecture but they will also to a great extend be about transforming the existing Architecture - to continue in the building analogy:

"...transform a skyscraper into a totally different skyscraper, while everyone residing in the first skyscraper still lives there."

[57]

The quote above is a good illustration of the challenge of projects not being Green Field projects. If we can master this discipline without disturbing the "residents", we have come a long way. SOA does promises to provide this possibility (see section 5), however I EA must be a part of this process to support an enterprise-wide transition.

The passages that I have discussed here demonstrate that it is not only the goals of EA and SOA that are similar, but that the similarities go much deeper:

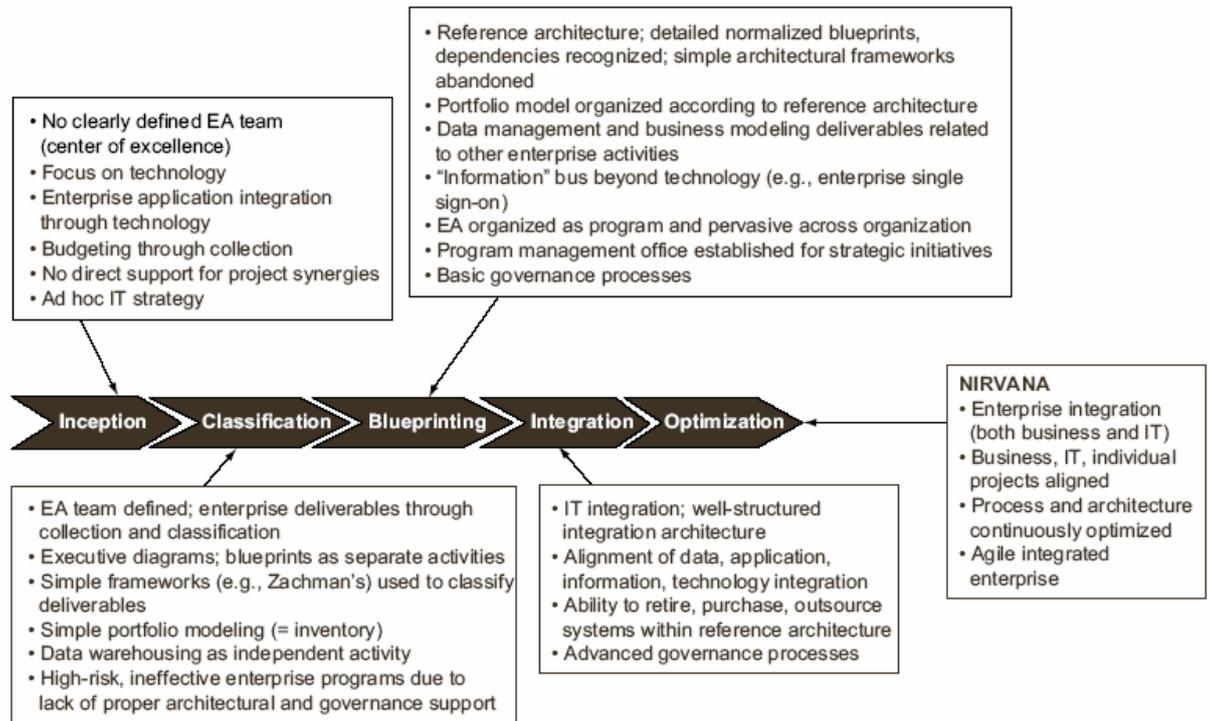
- EA and SOA both use an enterprise-wide approach on IT.
- The ERP solutions implemented throughout the nineties have some parallels with SOA. Let us use the experiences from these projects, and not repeat the same mistakes with SOA.
- Regardless of one is looking at EA or SOA there will be a legacy that must be taken into account when starting to transform the existing Architecture.

These similarities were also illustrated by the analogies used on EA also fits SOA. The key is in both EA and SOA are that they are enterprise-wide, meaning that the complexity level will be so high, that approaching either SOA or EA from a technical perspective is impossible. Furthermore the transition of having no enterprise-wide view on IT to having a full scale strategy in one step is also impossible. This must be done as a series steps, an issue which has been looked at with regards to EA, by looking at the maturity level of the enterprise.

7.5 Maturity of SOA and EA

In any case where one seeks to compare concepts, it is of the utmost importance that the concepts at hand are compared from a common base. In chapter 6 I defined my perception of the most important concepts related to SOA, but as I have discussed, these concepts are continually changing as they mature. The evolution is based on a better general understanding of the concepts, but the concepts will also mature within the individual enterprise.

Herzum have developed a Maturity Model for EA for which I see many parallels to how SOA will mature within the individual enterprise.



©2003 Herzum Software LLC. All rights reserved.

Figure 17 : Typical EA maturity phases. [Source: 57, p. 8]

Figure 17 is an illustration of Herzum's view on the maturity phases of EA consisting of 5 phases, and I will summarize these and discuss how they relate to SOA using the following method:

First I will describe the maturity phase as defined in Herzums maturity model.

This block will contain how SOA evolves within the individual enterprise.

I will follow the above with a short discussion of the relation between SOA and EA in the given level of maturity.



The **Inception phase** is the first maturity phase where EA activities will be focused on isolated projects, only with some effort on making enterprise-wide standards. The focus will often be oriented towards technology with a belief that technology can solve all problems. No EA-team exist, which means that no mechanism exist for supporting a long term view on EA [source 57, p. 8].

When commencing to implement SOA, it is often recommended to start of with a small scale pilot project the [46]. The focus of this project is to gain experience of technological issues, but also to gain experience on how the development process differs from ones normal procedures [87, p. 12][83].

Starting with a pilot project when working with new technologies or theories have always been the approach in software development. So this parallel can I my view not be seen as being a special case of either EA or SOA, but is just general practise. The interesting parallel is that the both EA and SOA starts of as projects focusing on technology, which essentially is in complete contrast with the recommendation of both EA and SOA - EA and SOA are not technical issues. One can then only speculate on why this is how EA and SOA are commonly approached. It will be my guess that it is partly in order to make the concepts more tangible, and secondly that it is the wrong people who are involved in the projects.

In the **Classification phase** an EA team is established and begins the foundation of a Reference Architecture. The focus will be the classification of what already exists. This should result in a high level view outlining the existing IT enabling the first step of an enterprise-wide management of IT - starting to view IT as a business. At this stage the EA team will often look at existing frameworks, such as the Zachman Framework, in order to classify the deliverables [source 57, p. 9].

After finishing the pilot project(s) a dedicated SOA-team should be formed to analyse the experiences of the previous phase. Their work will result in a Reference Architecture. This will unlike the EA Classification phase still be with a strong focus on the technical issues [87, p. 12][83].

The SOA-team will start looking after a supporting framework. Currently the only framework supporting SOA is the UDDI.

The establishment of a dedicated team is the key in both EA and SOA. Here EA has the advantage of being a more established discipline making it possible to take advantage of the experiences already made. EA will at this point start to elevate itself from only being a technical project, and look more at how IT can be managed on an enterprise-wide scale. SOA will still have a strong technical focus, looking at how the experiences from the pilot projects can be scaled into a bigger context.

What is not discussed here is who will man the teams, and what qualifications they should have? This will of course differ from enterprise to enterprise, but it is my guess that the EA team will only partly consist of people who were involved in the inception phase, whereas the SOA team probably will be almost identical to the existing SOA team. My grounds for these assumptions are based on my own experiences, however I see this as a very important reason behind the decoupling that is often seen between EA and SOA; the EA- and SOA teams will approach the vision of creating enterprise-wide IT very differently.



The **Blueprinting phase** is where EA is "lifted" from being an IT project to a strategic project with its own budget. The EA team will describe the "as-is" state to further detail and identify the "to-be" state. The EA initiative is now known and accepted in the enterprise, and both IT and strategic decisions will be aligned with cross-enterprise dependencies. The framework that was selected in the previous phase will now show inadequate and need modification or replacement. The first enterprise Services are released within the enterprise - however not widely used and needing to mature [source 57, p. 10].

The SOA-team should now have full support from the managerial level of the enterprise, including a budget and authority to enforce their policies and practices. The development of the first enterprise Services will need strong support from the SOA-team - if not fully managed and implemented by the SOA-team. This is fully aligned with the EA Blueprinting phase.

The work on creating an enterprise-wide SOA-data model, and in order to make this possible the SOA-initiative should be known and accepted across the enterprise.

A framework must be developed in order to support the state of the SOA. The UDDI is a technical framework and does not support the new strategic nature of the SOA. Since no such Framework currently exists, this must be developed.

When reaching the Blueprinting phase managerial support is required by both EA and SOA. The concepts are known throughout the enterprise, but the teams must help the actual projects - this is about "nurturing" the architecture.

The issue of Frameworks presents some interesting aspects. In EA the use of Frameworks was already introduced in the previous phase, and is in this phase maturing or being replaced. With regards to SOA the use of Frameworks is probably one of the biggest problems, as the only Framework currently available is the UDDI. The UDDI is primarily a technical framework for registering Services, lacking the ability to contain other architectural Artifacts.

The reason that no adequate Framework exists can be due to three reasons:

1. No enterprise has reached this level of maturity using SOA.
2. The enterprise keeps their Frameworks confidential.
3. The EA Frameworks in existence are adequate to do the job.

I will not elaborate on this issue here, but one thing is for sure; the UDDI is inadequate as an SOA Framework.



The **Integration phase** is of course about “integration”. The focus here are the IT aspects of integration, in order to ensure a cost effective communication between decentralised systems - some central systems might also exist. Ensuring enterprise-wide integration will improve the agility of IT in order to support the business.

Complexity reduction of the IT-portfolio is often commenced during this phase, which not only means looking at systems ready for retirement, but also aligning new purchases and outsourcing initiatives with the overall IT strategy. During work on reducing complexity great efforts will be made to identify common Services (or components) to ensure minimal replication of data and functionality.

The tools to support the EA will be taken a step further with better tool support to ensure direct “runtime” support of the EA [source 57, p. 12].

SOA is all about insuring seamless integration, so talking about an Integration phase in relation to SOA might be somewhat uncalled for. However, integration that scales need tools that can help ensure that the integration is enterprise-wide. As with the EA Integration phase, the SOA Integration phase is about minimizing the complexity in order to ensure a good and continuous agility and minimal replication of data and functionality.

This phase will also be where the business processes should be seen as a parallel to Services in the SOA - integration into the business [87, p. 12][83] - with a focus on classifying the existing systems with regards to retirement and further development.

All IT projects should comply with the rules defined by the SOA-team - all projects are now SOA projects.

In the Integration Phase EA is in fact focusing on the actual systems, e.g. moving from the general lines to technology. SOA is in fact doing the opposite and moves away from looking at the systems to looking at the business processes. An open question is if this is where EA and SOA correlate, and we get the desired link between IT and the business?

When the **Optimization phase** is reached the IT organization has all the architectural support, governance support and tooling required to manage IT as a business. Business processes are constantly being optimized, and the business is reaching significant levels of agility. The “to-be” state is realized and is no longer just an ideal future, and the monitoring is automated. Optimization is the main focus and new technologies that can improve the Return On Investment (ROI), are implemented quick and seamlessly [source 57, p. 13].

This is SOA! If the enterprise has desired to implement SOA it must be presumed that SOA is the targeted “to-be” state of the enterprise. It should be noted that Herzum is not aware of any enterprise that is at the Optimization phase yet.

If SOA is the Nirvana of EA, then it should be possible to use the definition of the Nirvana state of EA to define SOA by just replacing “EA” with “SOA”. The following definition is identical to that of the Optimization Phase of EA - the Nirvana of EA - replacing only the first four words with SOA:

SOA is reached when the IT organization has all the architectural support, governance support and tooling required to manage IT as a business. Business processes are constantly being optimized. The agile business is realized, and the monitoring is automated. **Optimization is the main focus** and new technologies, that can improve the Return On Investment (ROI), can be implemented quick and seamlessly.

Defining SOA again brings the attention back on the definitions of SOA I discussed in section 6.2, and interestingly all the definitions where focusing on setting the specifications on the



content of SOA - not what you should achieve by “going the SOA road”. This is exactly what the definition above does; defining when the goal is reached and not defining the goal itself.

Previously I have stated that the most important issue of SOA is agility, and now I am claiming that it is all about optimization. This is by far a contradiction as optimization is only feasible if the expense of the optimization doesn't exceed the possible gain. By increasing the agility one is also decreasing the cost of change; hence also optimization.

If the Optimization Phase defined by Herzum equals the goals of SOA I can't omit repeating the previously stated question; if I am confusing the concepts of EA and SOA? To which my answer is still; no. because the reason for this, is the differences in how “Nirvana” is reached of which the key issues are:

- The identification of the “as-is” state is one of the early tasks of EA, which was also discussed in section 6.2.2, but this is not an issue which is paid much attention in SOA. The early stage of SOA will typically start of with a small scale project [87, p. 12][83].
- Both EA and SOA focuses on the “to-be” state, but with a different focus; EA is focusing on the high level issues, whereas SOA is focusing primarily on the technical issues.
- EA is based on the existence of frameworks that supports EA. Currently there is little experience in using frameworks to support SOA.
- In the Integration phase the focus from the EA perspective is going towards the technical aspects, whereas from the SOA perspective the focus is going towards the higher levels.

The differences and similarities of EA and SOA show that they are not identical, but they are not like chalk and cheese either! However, as stated previously, it is clear that the relations between SOA and EA are immanently connected with the maturity level of both; It will make little sense to look at the relations between EA in the Optimization phase and SOA in the Inception phase.

7.6 Summary

I started out to identify where SOA fits in the evolution of software development paradigms, and from here I looked at the concept of evolution in general. Here I found the work of Geoffrey A. Moore as a good frame of reference to illustrate that evolution is closely tied to maturity. I then used Jeff Schneider to demonstrate that the concept of SOA has gone through an evolution, and matured as a concept.

Since my theory is that SOA is closely related to EA, I found it relevant to identify some general parallels of EA and SOA. I did this using Herzum's work identifying three main parallels of EA and SOA: they both see IT as enterprise-wide, there is similar use of the analogies and they will not be Green Field projects. Many other common aspects could be identified but the purpose was to set the scene for the main issue of this chapter; that comparing EA and SOA with no common reference to maturity is not possible.

Herzum's Maturity Model of EA showed that the phases of EA could be seen in parallel with the maturity phases of SOA. This process resulted in the key finding of this chapter: The definition of the most mature state of EA equals the goal of SOA - SOA is the Nirvana of EA.

At the time of writing I was not able to identify sources to directly support my perception on how SOA is related to Herzum's Maturity Model. The work was partly based on indications identified in different sources, but was mainly based on my own experience. As it happens an article has now been published defining the following SOA Maturity Model (SOA MM):

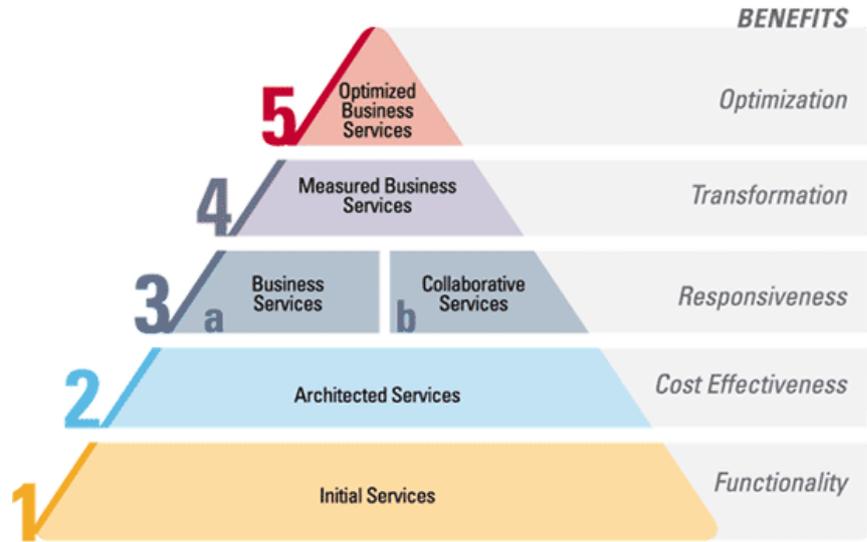


Figure 18 : SOA Maturity Model Levels with Key Business Impact [Source: 129]

The SOA MM illustrated Figure 18 is a 99% match of how I described the maturity levels of SOA in section 7.5. The article does not look at the relation between SOA and EA, but is backing up my perception on SOA maturity.



8 Perspectives of SOA's impact on EA

The reasons for starting on an EA program may be many, but the main motivation will often be to reduce redundancies, minimize the creation of silos, and increase the ability to make informed managerial decisions - all which essentially are founded from economical reasons. What EA should provide is a common, shared blueprint for the business and IT [23]. Looking at the business IT from this perspective is clearly a top-down view.

Both EA and SOA have evolved and matured, and as I discussed in the previous chapter the goal of EA at its most mature state equals that of SOA. The common goals of EA and SOA, such as creating interoperability, reuse and eliminate duplication can to some level be reached running a "clean" SOA-project. But the true value lies in combining EA and SOA - together they can guide interoperability, not only between IT systems but between the business and IT - Enterprise-wide interoperability [23].

To further explore the relations between EA and SOA I will in this chapter approach the issue using the different perspectives on the ideal EA and SOA, e.g. the Nirvana state.

In October 2003 Boris Lublinsky and Dmitry Tyomkin in their article "Dissecting Service-Oriented Architectures" look at how SOA impact on EA [89, p. 55]. They based their method on that EA can be viewed from four perspectives [89, p. 53][90], and I will use this as the base for the rest of this chapter.

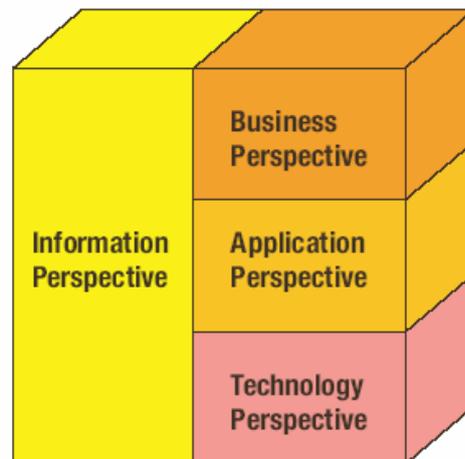


Figure 19 : Views of Enterprise Architecture [Source: 89, p. 54]

From these four perspectives they set out to describe how SOA will impact EA. The scope of their article unfortunately does not allow them to go into much depth with the subject, but it does touch upon some interesting issues which I will discuss in the following.

8.1 Business perspective

From a business perspective SOA will help connecting the processes of the business into IT, which will make it possible to easy to implement the changes from the business into the IT level. The connection between the business and IT is focused on the identification of Services [89, p. 55].

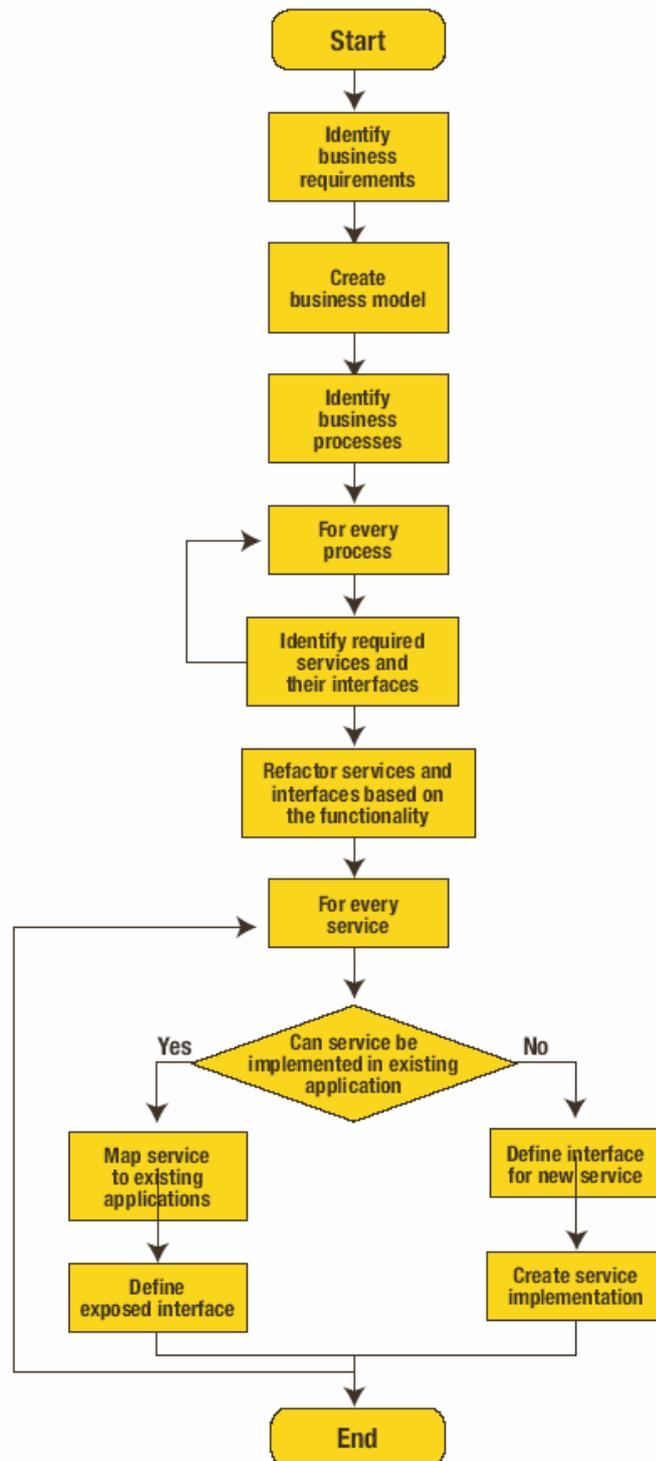


Figure 20 : Defining Business Services [Source: 89, p. 56]

The model illustrates a series of steps, starting with identification of the fundamental processes of the business. The next steps should then map these processes into Services and identify if these are supported by the existing IT-portfolio, or if new development is required [89, p. 55].

As I noted earlier, they did not go into much depth in their work, but the model they have formulated for defining the business Services is a good example on how SOA affects EA. Not only is it necessary for the enterprise to set their minds to work with business processes as Services, but the model is a good example of an EA-Artifact.

8.2 Application perspective

From an application perspective the big change when “going SOA” is that Services entail loose coupling. This does not only make it possible to develop each Service independently, but allows a new approach to maintain the application portfolio.

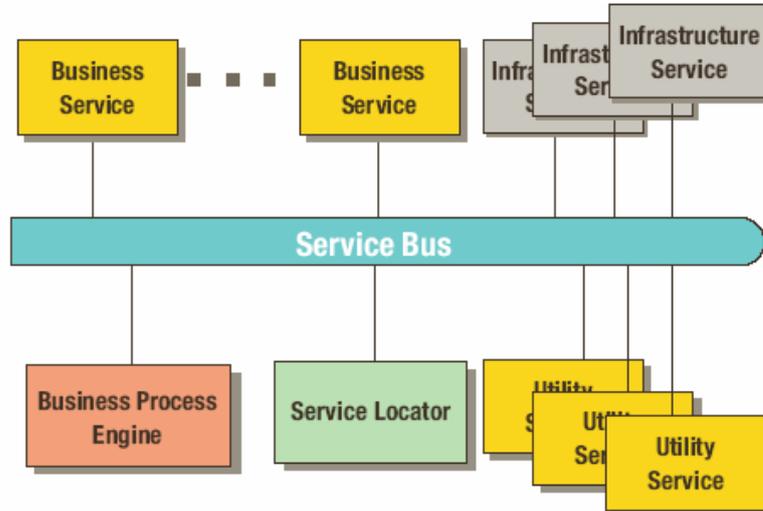


Figure 21 : Typical SOA Application Architecture [Source: 89, p 57]

Figure 21 shows the typical application architecture of SOA. The key issue in terms of easing the management of the application portfolio is in fact not the Business Services, even though the Business Services are the “content” of the application portfolio. It is the possibility to extract the general architectural aspects and implement these in Services. In Figure 21 these are identified as; Business Process Engine, Service Locator, Utility Services and Infrastructure Services [89, p. 56].

The very concrete look at what Boris Lublinsky and Dmitry Tyomkin see as Typical SOA Application Architecture (illustrated in Figure 21) is for sure relevant to define for the individual enterprise, as this is a concept that have many definitions - such as Enterprise Service Bus (ESB) [11, p. 5-6][10, p 16] [2, p. 74] [30]. I will not at this point go into the discussion of what an ESB is, however interesting. It is the more general issue of looking at the Application Architecture as a part of the EA that is interesting. Why is this EA? First of all it is enterprise-wide, and secondly since no definition of an SOA should be implemented (the SOA Application Architecture) this must be done at an very early stage of the SOA process. Making this conceptual model of the SOA Application Architecture is also an EA Artifact.

8.3 Information perspective

Processing information is essentially what IT is all about, and introducing SOA makes information an even more central issue. Using the word “central” is in fact indicating what is added to the “normal” way of viewing information.

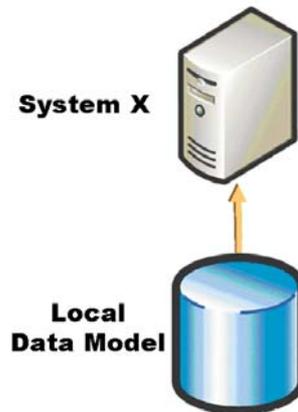


Figure 22 : Normal information view [Source: own work]

Figure 22 is a simple illustration of how information is viewed in most systems today; it is only the individual systems that know about the given data model. This is as such not a problem, if you don't have to integrate with other systems. However, it will be impossible to talk about SOA and not talk about integration, so this way of handling data will not be applicable in an SOA.

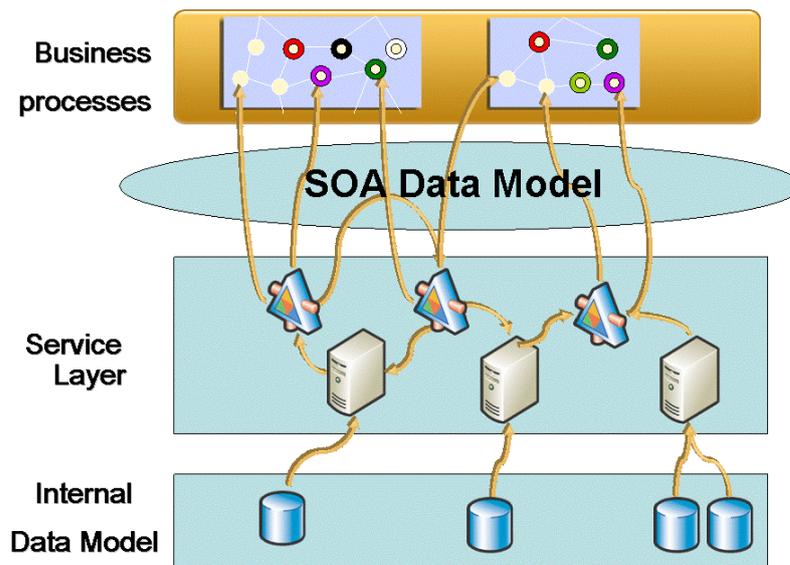


Figure 23 : SOA information view [Source: own work]

Figure 23 shows the introduction of the “SOA Data Model”, or as Lublinsky and Tyomkin calls it: “Data dictionary for Service messages, defining communications semantics of the SOA”. What is important to note is that the underlying systems are still using their own internal data model source 89, p 56].

As with the Application perspective, Lublinsky and Tyomkin are actually making a choice of architecture. In this case it is founded in the information architecture, but will have an immense impact on the overall application architecture. The strategy chosen is in fact similar to the “Connect Strategy” formulated by Gartner [91][Appendix E]. So what Lublinsky and Tyomkin are doing is making a choice - an example of how SOA can impact EA. The Connect Strategy is also by itself an EA Artifact.

8.4 Technology perspective

The Technology perspective is essentially about implementing the model developed in the Application perspective (see section 8.2). But in order to implement these aspects of SOA it is usually required to develop an infrastructure which is shared between the Services.

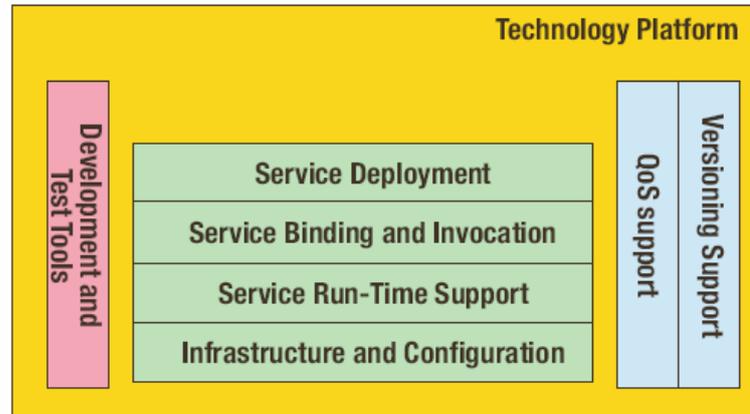


Figure 24 : Conceptual View of Technology Architecture [Source: 89, p 57]

Figure 24 is a conceptual view of the technical architecture, and is unfortunately not described in much detail. The following is quoted from the report of Lublinsky and Tyomkin [89, p 57]:

- **Service deployment:** concerns the processes and technology choices around deployment of Services, including host platform
- **Infrastructure and configuration:** provides middleware, operating system, hardware, storage, networking, and the trust and management support for the whole system
- **Service run-time support:** hosts the process, logic, functions, and state management required by a Service based application and is the full enterprise application environment with specific support for Services
- **Service binding and invocation:** contains Services binding and invocation mechanisms, including support for both locating and invoking enterprise Services and exposing applications or code as Services in different operational environments.
- The run-time component is responsible for the additional support for SOA and consists of two major parts:
 - **Quality of Service (QoS) support**
 - **Versioning support.**

As previously discussed, the Technology Perspective is about implementing the model developed in the Application perspective. At the Application perspective it was the Business Process Engine, Service Locator, Utility Services and Infrastructure Services which were in focus, but at the Technology Perspective there is much more focus on the Service Bus. Unfortunately Lublinsky and Tyomkin don't elaborate much on the Technology perspective, and which parts of the Application perspective corresponds to the parts of the Technical perspective, and how these issues are practically implemented. However, this is not currently the issue of concern - it is how this will impact EA. In my view it is obvious that there is the need of a documentation Framework, which is a big part of EA. The Conceptual View of Technology Architecture is an EA Artifact, but it will have little value if the relations between the Application perspective and the Technology perspective aren't traceable.

8.5 Summary

My motivation behind this chapter was to support my theory that SOA will change EA. I based this on the work of Lublinsky and Tyomkin who are looking at SOA's impact on EA, but we have only discovered the top of the iceberg! In each of the EA perspectives they identify an EA Artifact which is a direct result of SOA, but what is not discussed, is how these Artifacts



are related to one another. I have pointed out that the relation between the Application perspective and the Technology perspective aren't traceable, but what puzzles me even more is where the Data dictionary, defined in the Information perspective, fits in the grand picture. In Figure 19 the Information perspective is depicted as crossing all other perspectives, which would indicate that relations would exist - shouldn't the Data dictionary be a part of the Application Architecture? As an example of where the data dictionary is a central part of the Application Architecture is the Danish eGovernment initiative, where the Infostructurebase's primary goal is to enable interoperability between systems [99].



9 Service Oriented Enterprise Architecture

Until this point the goal has been to identify that there are parallels between EA and SOA. We have seen that there are parallels on several different aspects:

- Evolution (chapter 7)
- Maturity (chapter 7)
- Perspectives (chapter 8)

But what still remains to be answered is; what is the consequence of this? If SOA and EA aren't the same but have so many crossing elements I see three possible scenarios of how SOA and EA will evolve:

1. EA is absorbed by SOA
2. SOA is absorbed by EA
3. EA and SOA will merge into a new concept

First I will rule out scenario 1 even though this might often be what will happen as a result of seeing SOA as the "silver bullet". My reasoning for this is based on an interview I did with Mikkel Haugsted Brahm who is Chief Consultant of Development and IT in Nykredit³³. They were implementing SOA, and it was not discussed as an EA project, but their SOA program went on having characteristics of an EA program [Appendix E]. This development, I predict, will give them a great deal of problems, and I will return to this interview later to elaborate.

Scenario 2 and 3 are not necessarily different, but this is where I see the steps of evolution are moving. I have in the previous sections showed that SOA will influence EA in many ways, but if this results in changes big enough to talk about a new concept I will currently leave as an open question. The problem is to determine when a concept has changed enough to qualify as a new concept!

I have titled this chapter Service Oriented Enterprise Architecture (SOEA) which of course is an indication that I see EA and SOA as merging into a new concept. The reason that I have chosen the term SOEA is done based on the following equation:

$$\text{SOA} - \text{A} + \text{EA} = \text{SOEA}$$

I remove the "A" from SOA because I don't see it as the same "A" that is in EA. As I stated in section 6.2: The A in SOA stands for architecture, and this is of course also a central aspect of SOA, but what really matters is the business design and delivery process [21, p. 4][37]. Another reason for removing the "A" from SOA is that I see it as the main reason of the confusion about whether or not SOA is a technical issue or not. However the "A" in EA has a more general meaning and is not directly associated to technology. The reason for including "E" (Enterprise) is because the term enterprise indicates "enterprise-wide", which is exactly what is needed to harvest the advantages of SOA.

"It (red. EA) covers all aspects of IT, including user interfaces, portals, new and legacy applications, technical infrastructures, and databases. And, over time, it should address the whole software supply chain, including maintenance, evolution, retirement, distribution, and the operation of applications inside the enterprise and across the virtual enterprise."

[57, p. 3]

³³ One of Denmark's leading financial services providers, and also one of Denmark's leading SOA enterprises.



The quote cited above says exactly how I see the real challenges of SOA. In section 5 I listed some of many sales pitches that have been made of what SOA will give. Most of them which remains to be proven, but where I see the main problem is that SOA is sold of as a magic remedy that bring you all the promised advantages almost effortlessly. It is my opinion that you couldn't be more wrong. If SOA is to be a success, methodologies must be developed to help coping with the challenges pointed out in the quote above, and bring you closer to the "Nirvana" state of maturity (see section 7.5). However, before it is possible to develop any methodologies it is necessary to grasp the problem. I will not try to develop a methodology as this is out of scope of this thesis; the purpose of this thesis is to shed some light on the problem which then can be used as the base for developing a methodology in future work.

I have until now shown a series of examples on how SOA influences and change EA on a conceptual level, but my motivation for writing this thesis is very much founded in a desire to make the relation between EA and SOA as concrete as possible, which has proven to be quite difficult with fuzzy concepts as EA and SOA. Initially my thoughts were to develop a common Framework for EA and SOA, based on the Zachman Framework. This proved not to make the relations much more concrete, as the Artifacts of such a Framework would solely be based on the experiences of EA. But, the idea of using the concept of Artifacts to identify concrete relations between SOA and EA was indeed appealing and will be the foundation of this chapter.

9.1 SOA Artifacts

As discussed in section 6.1.3; EA will create EA Artifacts [1, p. 111]. EA Artifacts are a well known concept and is the foundation of the EA process, and EA as such. But what about SOA, will SOA produce any Artifacts only relevant for SOA, or will Artifacts produced by SOA be EA Artifacts?

The concept of a SOA Artifact is not a known concept - yet. There are however a couple of opinions on the idea. The simple notion is equating the Services of a SOA with Artifacts [92], and I agree that they should be perceived as such. There are also examples of Artifacts being created without the creator knowing that they are SOA Artifacts. An example of this is the OASIS' Tax XML TC who set out to analyse personal- and business tax reporting and compliance information to facilitate interoperability using XML. But more importantly they were also chartered to produce a repository of Artifacts including XML templates, vocabulary of terms, documents exchanged for tax compliance, best practices, guidelines and recommendations for practical implementation [93]. The work was not as such initiated with a focus on creating SOA, but just as a good documentation model. They do however note that the work can be used in an SOA, indicating that this is a central part of insuring the interoperability issues of SOA. A last indication of SOA creating Artifacts has a direct parallel to the Zachman Framework; creating a framework for SOA such as David Sprott and Lawrence Wilkes in their article "Enterprise Framework for SOA".

Conventional enterprise architecture describes an information system in terms of structural properties of the system. The architecture identifies components, building blocks, standards, policies and products which form the basis for planning and guiding systems delivery.

Not surprisingly SOA introduces change to the structural properties. There are new and different building blocks, standards etc. These don't necessarily replace the existing properties, mostly they complement and extend. However there are also areas where fundamental differences apply, for example in areas such as scoping and applicability, security models, reuse policies and so on.

[46]



Sprott and Wilkes do not directly use the term “SOA Artifacts”, but they support my notion that SOA will change EA, and in some cases add completely new issues to EA. But how will the change show in practise?

As stated in the previous section it is difficult to develop a methodology to a problem that is not known. This is where I see the identification of the SOA Artifacts that typically are needed in order to create a successful SOA. Before identifying any SOA Artifacts I will look at what a SOA Artifact can be.

The definition for the term; Artifact:

“Any manually portable product of human workmanship. In its broadest sense includes tools, weapons, ceremonial items, art objects, all industrial waste, and all floral and faunal remains modified by human activity - any physical remains of human activity.”

[53]

I like this definition as it clearly shows how general the term is; anything human made (the antonym being: natural object). This roughly means that anything created in the context of SOA will be an SOA Artifact. To narrow down the result set I will base this issue on how this is approached in EA. Here the concept originates from Zachman, as the filling of his Zachman Framework [94], which in more concrete terms means any kind of representation, model or diagram that are easily understood by business people [95, p. 1368].

That EA Artifacts must be understood by business people is still a very broad definition, and the question is if this also applies for SOA Artifacts? I believe that the answer is both yes and no, and there will be SOA Artifacts that the business people must understand, but there will also be SOA Artifacts they will not understand. In fact the same issue could be raised with regards to EA. However it is of the utmost importance that all Artifacts are “tied” to a high level SOA Artifact that originates in the business. This is in my opinion also the case for EA Artifacts, and I don’t agree with Carla Marques Pereira and Pedro Sousa on this [95, p. 1368] - their picture is more fine-grained than they claim.

9.2 Identifying SOA Artifacts

Artifacts are usually identified through a process of analysing a specific case and will also be a result of the maturity level of the enterprise. However my goal is somewhat different than this as I am looking at how SOA will influence EA - not a specific case. The focus of the Artifacts should therefore be neutral of a given case and be examples of general Artifacts that are needed to make SOA a success. In fact I have already identified several SOA Artifacts in the previous sections. In section 6.2.4 I stated my view on how a Web Service is defined, and ended up with a more elaborate definition than that provided by W3C. The main reason for seeing the W3C definition to vague was that in an SOA context is a Web Service not just a Web Service - an example of making a context less concept into context. Another example of SOA Artifact candidates are the EA Artifacts identified in the previous chapter:

- Defining Business Services
- SOA Application Architecture
- SOA information view
- Conceptual View of Technology Architecture

I will not use these Artifacts directly as they are not documented to a level on which I can make any qualified discussion directly based on these.

My motivation of this thesis is based in my own experiences of SOA during the last couple of years, which will also be a big part of the base of this section. My thesis problem focuses on how to manage the evolution of an SOA, and as I have discussed in section 7.2; evolution



entails change. So in order to identify areas of interest I have here listed some of the causes of change:

The technical needs can be:

- new Services are added
- Services are being removed
- new Services are created from existing Services.
- Services are changed (not the interface, but the system behind)
- The interface to a Service is changed.
- The location of a Service changes.
- Integration between to separate SOA systems.
- Date modelling
- Integration to legacy systems

The business needs can be:

- New products (especially in the financial world such as a new mortgage type)
- New requirements to BPR
- Merging of two departments
- Movement of tasks to another/new department (or even outsourcing)
- Financial responsibilities moved from one department to another

I will use the following model to make a general description of potential SOA Artifacts that in my view are needed to manage the changes caused by the issues listed above. The purpose of these descriptions is to make the foundation for selecting a set of these on which I will elaborate further.

Title	A short title of the SOA Artifact
Description	A short description of the SOA Artifact
Potential derived Artifact(s)	Potential SOA Artifacts derived from the high level Artifact. These can then again derive to other Artifacts, but in order to stay within the scope of this thesis I will only briefly describe the SOA Artifacts derived from the high level SOA Artifacts.
Relation to EA	If there are relations that must be planned as a conjunctive effort between EA and SOA.

Table 6 : Documentation model for SOA Artifacts [Source: own work]

As previously noted I see that the high level Artifacts will form the foundation for the more concrete Artifacts that are needed to make the architecture operational - this is where it is possible to make the connection between the business and the technical aspect of IT. My main focus will be on the high level SOA Artifacts and how these relate to EA. This relation is a general view, and is not connected to any specific EA Artifact. The high level SOA Artifacts I have identified are:

- Definition of SOA team
- Defining SOA concepts
- Documentation Framework
- Defining the goals for SOA (SOA Vision)
- Making the business think SOA
- SOA adoption roadmap
- Return of Investment Plan.
- Pricing Policy Model (PPM)
- Data Model Policy



- SOA Governance
- Partner Integration Policy
- Quality control
- Develop Service Oriented Development (SOAD) method
- Enterprise Service Bus Policy
- Use of standards

Title	Definition of SOA team
Description	Introducing SOA is not just another project; it will probably be the most challenging project the enterprise has ever initiated [96], so having a dedicated team of highly qualified people is critical.
Potential derived artifact(s)	<p>Definition of authority A SOA team with no authority will have little luck in transforming architecture and even more challenging is the transformation of the business. The team must therefore be equipped with a formal, and publicly documented, authority.</p>
Relation to EA	If an EA team exists there will be no need to form a SOA team - as it already exists. By this I am referring to that the level of cooperation between an SOA- and an EA team will be so important that it would be dangerous to have two separate teams. Instead I would recommend that the needed resources should be added to the existing team. What the team is called is of less importance.

Table 7 : SOA Artifact # 1

Title	Defining SOA concepts
Description	<p>When “going SOA”, a goal should be to break down the gaps that often arise between the formulation of an EA to the actual implementation of the EA in IT, meaning that people with very different backgrounds will be working together. This has of course always been done, but often there is a clear distinction for when a task is moved from one level to another. This makes it possible to create a formal way to pass the task on using a tool such as UML³⁴. In an SOA the distinct boundaries between different levels are broken down, and all levels of the organisation should “talk” SOA.</p> <p>This fact makes it essential to ensure that the whole organisation have the <u>same</u> comprehension of the concepts of SOA in order to share a common goal.</p> <p>This is however not as easy as one might perceive. SOA, and the concepts around SOA have been around for some years, and one might expect that the concepts should be well known and defined, but nothing could be further from the truth [34].</p> <p>One of the main problems is that SOA spreads across many levels in an organisation, hence, a lot of people with different backgrounds will work with SOA, and as a consequence, a lot of diverting definitions of SOA will arise [26]. As a parallel we can look back at the work of Zachman from 1987</p>

³⁴ Unified Modeling Language™ (UML) www.omg.org

	If you are:	Then you probably think architecture is:
	A programmer	A structure chart
	The database administrator	Data design
	An analyst	A data flow diagram
	A planner	Some combination of entity/relationship diagrams and/or functional flow diagrams
	The communications manager	The business logistics infrastructure and/or the distributed systems architecture
	An operations manager	The system architecture
	A network administrator	The network architecture
	A program support representative	Detailed data and program descriptions
	A computer designer	Machine language (not represented on the summary chart, Figure 2)
	The president	Entity classes, process classes and/or a map
	<p>Figure 25: If/then view of architecture [Source : 65]</p> <p>This added with the natural maturing of both the conceptual and technical aspects around SOA makes the picture of what SOA is even more blurry.</p> <p>SOA itself uses a lot of concepts that are prone to the same definition problem as SOA itself - hence the title of the section in plural "Defining SOA Concepts". All central concepts that are relevant to the organisations definition of SOA should be defined in a publicly accessible place!</p> <p>A common understanding of concepts is always a prerequisite of working together and is by no means special for SOA.</p>	
Potential derived artifact(s)	<p>Reference Model</p> <p>A reference model can be used to support the continued evolution of a concept, such as SOA, while keeping on a common platform of reference.</p> <p>An example of such an initiative is currently being developed by OASIS in their "SOA Reference Model TC³⁵".</p>	
Relation to EA	<p>Defining concepts is, as discussed, not a special case for SOA, but there is however a strong relation to EA as some of the concepts defined in the light of SOA might change the concepts of an existing EA [46].</p>	

Table 8 : SOA Artifact # 2

³⁵ http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm



Title	Documentation Framework
Description	<p>Defining the SOA Artifacts is part of the documentation of SOA, but documenting it without having a model for arranging the documentation will in most cases diminish the value of the documentation - it is seldom the individual piece of documentation that has value - it is the relations between them that holds the value.</p> <p>The adoption or development of a Documentation Framework is therefore of the utmost importance. An example of such a Framework has been developed by David Sprott [46]. This should only be seen as an initial attempt, and approaches the challenge of managing the relations.</p>
Potential derived artifact(s)	<p>Service map A Service Map is a “map” that identifies SOA business Services candidates [100] [Appendix E]. It is an Artifact to be placed within the Documentation Framework, but it will also by itself work as a sub-Documentation Framework.</p>
Relation to EA	<p>The Documentation Framework is one of the most central aspects of EA, and since SOA will affect a lot of the issues normally covered by EA, it must be analysed if there are any crossovers between the SOA Framework and the EA Framework - It is even possible that the two should merge into one.</p>

Table 9 : SOA Artifact # 3

Title	Defining the goals for SOA (SOA Vision)
Description	<p>Any project without a specified goal is almost sure to fail, which is also the case for SOA. Since SOA can provide advantages to many levels of an organisation it is important to define a main goal - are you updating your technical platform or are you changing your business?</p> <p>An example of such a Vision is that of Danske Bank³⁶ which have the vision of; one bank, one system” [101]. A vision I believe is somewhere in between of being technical- or business goal, but the power of such a goal is that it is understandable at all levels of the business.</p>
Potential derived artifact(s)	<p>Scoping [46] Defining goals is often done as a “one-liner” to ensure that everyone can remember the vision of the enterprise. But in order to detail the vision into a more operational concept it is important to identify the size of the task at hand - what is the scope.</p>
Relation to EA	<p>The example of Danske Bank clearly illustrates that this is an EA issue. The vision of SOA is basically how the enterprise is to work with IT.</p>

Table 10 : SOA Artifact # 4

³⁶ Danske Bank is the largest bank in Denmark and a leading player in the Scandinavian financial markets (<http://www.danskebank.com/About>).



Title	Making the business think SOA
Description	The value of SOA is not a technical issue; the real value comes from making the entire business think Service Oriented [102]. IT people cannot make the right Services for the business - they can only make the infrastructure [102].
Potential derived artifact(s)	Education Making the business respect SOA can only be done through information.
Relation to EA	SOA must not be seen as a replacement for EA, hence the coordination between EA and SOA have to be in place before bringing SOA to the entire enterprise.

Table 11 : SOA Artifact # 5

Title	SOA adoption roadmap
Description	Incremental integration is one of the sales pitches of SOA that goes well with the fact that SOA projects almost never are Green Field projects - however a roadmap for the implementation must be made to support the strategic direction and detailed plans for governing the implementation of change of both business design and SOA[104]. A SOA is not a “big bang” project, but an incremental implementation process [15, p.5]. SOA can be seen as an evolution of the existing systems by breaking down barriers between existing systems [18, p. 4]. The projects must of cause start their technical implementation at some point which can be done with a big or a small “bang”. An example of a “big bang” project is Danske Bank who merged with BG Bank ³⁷ . Here the systems of the two banks were “reduced” to one system over a weekend as a result of 700 IT-people’s work for seven months. There can be no general approach for at SOA Adoption Roadmap, as no general definition of SOA exist, plus it is depended on the legacy systems of the enterprise.
Potential derived artifact(s)	“As is” analysis What is the current system portfolio? “To be” analysis What is the desired “To be” state.
Relation to EA	Making a radical change like “going SOA” can’t be done without a strong coupling to the EA. The two derived products are generally seen as EA products, but if the decision is to “go SOA” this will evidently be the result of the “to be” analysis.

Table 12 : SOA Artifact # 6

³⁷ A Danish bank



Title	Return of Investment Plan.
Description	The time perspective of ROI in SOA can vary greatly. A long term ROI will often give the most agile system, whereas short term ROI will give a less agile system [35]. It is important to have a strategy on how the company wants to level the time span of ROI vs. system agility.
Potential derived artifact(s)	Specify ROI Setup measurements that can verify that the expected ROI has been achieved.
Relation to EA	The reason for commencing EA is mainly to increase profits (see section 6.1.1.1)

Table 13 : SOA Artifact # 7

Title	Pricing Policy Model (PPM)
Description	<p>Normally integration between systems is not a desired event. Integration is however a very wellknown issue in almost any company - new demands on the business requires the systems to interact in new or changed ways, and integration-work always comes at an economical cost.</p> <p>In the case of integration between two systems the billing of the expenses is usually straight forward; if you are going to use functionality/retrieving data from my system, you will have to pay. SOA is all about integrating systems³⁸, however not on a “one to one” basis, but through opening systems using public³⁹ Services. Often there will be no direct economic incentive for the Service provider to share data and business functionality.</p> <p>An example of this is the Danish Central Business Registration (CVR) who sells data through a Web Service. CVR has made their data available for others through a Web Service. Their price model is⁴⁰:</p> <p>Price pr. unit:</p> <ul style="list-style-type: none"> • Level I: CVR-nr., P-nr., name and address 0,35 Dkr. • Level II: Level I + trade, type of enterprise 0,75 Dkr. • Level III: Level II + phone information 1,00 Dkr. <p>Connection fee pr. company: 5.000,00 Dkr. The collected purchases of CVR-online data can not excide 750.000 Dkr. pr. year.</p> <p>The price model of CVR is in my view a new line of business for CVR themselves - a new source of income. In my opinion the price model chosen by CVR does not comply with SOA, it is just a new product published using Web Service technology. This view is also an illustration of the different views that exist on SOA, as the CVR-example in fact is</p>

³⁸ One of the main points made by Mark Colan: Innovation Evangelist IBM Emerging Technologies on the Software Development Forum 2005 in Copenhagen [SOA - So What?].

³⁹ Public in this case can mean public to the business itself - on the local network.

⁴⁰ Translated from the CVR homepage <http://cvr.dk/cvr-online.html> 25 March 2005



	<p>promoted as SOA in the book “Service Oriented Architecture - Integration as a competitive parameter⁴¹” [122].</p> <p>This approach is not liable to work in an SOA. Service reuse will have little success if not all parties involved benefit, resilience is sure to arise. In order to share the benefits of having many applications using the same Service the following must be achieved:</p> <ul style="list-style-type: none"> • the cost should be shared across all the involved parties [19, p. 7] or, • budget control should be supported from a central unit in the organisation, compensating the expenses of the Service. <p>The key issue is; the agility of SOA mustn't be inhibited by economics. Services are to be used as much as possible, and the cost should be shared for mutual benefit. For this to become reality a common Pricing Policy Model (PPM) must be developed.</p> <p>If a PPM is not developed, the risk is that Services will not be reused and data will be duplicated across the SOA. At best the development will be delayed - and more expensive - due to time-consuming economical negotiations of every negotiation. The consequences of not creating and govern a PPM is illustrated by how data from the Danish Central Office of Civil Registration (CPR) was resold:</p> <p style="padding-left: 40px;">CPR is selling information about the Danish citizens to private companies. The price model is somewhat similar to that of CVR - you pay according to use. What happened in the CPR-case was that several companies⁴² bought information from CPR and some of these⁴³ companies resold these. The consequence was that information was duplicated in an uncontrolled fashion. The primary reason being the PPM defined by CPR did not fulfil the demands of an SOA.</p> <p>Distributing data by copying is a violation of the “data once, use everywhere”. It is therefore important to develop a PPM that removes the motivation to copy data for economical reasons. It should be noted that the rule of “data once, use everywhere” is a conceptual rule and does not imply that the physical implementation duplicate for performance or security issues.</p>
<p>Potential derived artifact(s)</p>	<p>PPM standard To provide the full agility of SOA the PPM should be implemented in a way so that the system itself can take care of the billing process automatically.</p> <p>Setup PPM staff It would be naive to believe that billing process can be developed in “one take”, plus that it will not be subject to conflicts and change. To take care of these issues a staff must be given the proper authority to solve conflicts and further develop the PPM.</p>

⁴¹ Written in Danish - title is translated.

⁴² <http://nyhederne.tv2.dk/baggrund/article.php?id=2108774>

⁴³ <http://nyhederne.tv2.dk/baggrund/article.php?id=2107412>



Relation to EA	Who is to pay, and who is to gain? When asking this question in the scope of the entire enterprise the answer is; this is an EA issue.
----------------	--

Table 14 : SOA Artifact # 8

Title	Data Model Policy
Description	This issue is described in section 8.3
Potential derived artifact(s)	<p>Identifying data owners In many enterprises there are no clear definitions of who owns data. An example could be customers. A customer can exist in many forms within an enterprise, resulting in many different data owners. This was also the case in Nykredit. This is generally a problem if one wants to have a holistic view of a customer's engagements. Nykredit have approached this issue and created a common model for the customer-concept within the enterprise. This raised a lot of problems, but it is of eminent importance to create these policies to ensure that data are trusted and not duplicated.</p> <p>SOA Data Model From the foundation of the Data Model Policy an actual SOA Data Model (see section 8.3) must be developed at both conceptual and logical level.</p> <p>Data Model Staff The SOA Data model will not be consistent and will require constant development and governance.</p>
Relation to EA	SOA changes the level of abstraction on the data modelling work. It is no longer a "system issue" but a matter of defining the concepts of the enterprise. Data is a central part in the Zachman Framework, and introducing SOA will change how to approach this issue.

Table 15 : SOA Artifact # 9

Title	SOA Governance
<p>Description</p>	<p>Governing an SOA is not just a one-off exercise, it is a continuous process consisting of several elements as illustrated below by Richard Veryard.</p> <div data-bbox="496 398 1442 1173" style="border: 1px solid black; padding: 10px;"> <p>The diagram illustrates SOA Governance. At the center is a pink box labeled 'Network of Services'. Above it are two white boxes: 'Business Internal' and 'Business External', with arrows pointing down to the network. Below the network are two white boxes: 'Software Internal' and 'Software External', with arrows pointing up to the network. Surrounding the central elements are four blue double-headed arrows: 'Business Governance' at the top, 'Software Governance' at the bottom, 'Service Governance' on the left (vertical), and 'Business Governance' on the right (vertical).</p> <div style="display: flex; justify-content: space-between;"> <div style="width: 60%;"> <p>Software Governance Coordinating software development, acquisition and (re)use across internal and external domains to achieve maximum agility and economics of scale/scope. Monitoring the technical performance of software services, including security.</p> <p>Business Governance Coordinating business development, negotiation and collaboration across internal and external organizations, including trust. Monitoring the business performance of services, including ROI.</p> <p>Service Governance Aligning software governance with business governance.</p> </div> <div style="width: 35%; text-align: center;"> <p>Figure 26 : SOA Governance [Source: 104]</p> <p>Potentially an organisation can have thousands or even hundred of thousands Services. In order to prevent chaos it is important to create both means and ways to maintain control. Maintaining control of the Services is only one part of the challenge; the real challenge is to keep the business aligned to the Services, and vice versa - illustrated in Figure</p> </div> </div> </div>
<p>Potential derived artifact(s)</p>	<p>Service choreography standard BPEL⁴⁴ have almost become the default choice for choreographing of Services. But BPEL is a programmer's tool and must be supported with a methodology [105].</p> <p>Service Ontology [46] Having consensus on what a Service is, is not enough. Services will exist at many levels and it is necessary to support this diversity of Services in a unanimous way. CBDI suggests that a service ontology can consist of the following layers of Services: Process Service, Core Business Service, Underlying Service, Utility Service and Infrastructure Service. The number of layers should be adapted to the individual enterprise [46].</p>
<p>Relation to EA</p>	<p>EA and Governance are often seen as parallels as it is the symbioses of these that make EA realistic over time. SOA Governance is a part of the EA Governance as illustrated on Figure 26 - Business Governance is certainly a part of EA Governance.</p>

Table 16 : SOA Artifact # 10

⁴⁴ Business Process Execution Language.



Title	Partner Integration Policy
Description	<p>In order to take advantage of SOA you must integrate your system with your partners; supply chain, external sales channels etc. [40] When integrating with new partners it is important that this is done in a controlled and documented fashion. In SOA the interface of the Service is the technical contract but there are also many other concerns that need attention.</p> <p>Generally the integration strategy can be divided in:</p> <ul style="list-style-type: none"> • Negotiated - when provider and consumer negotiate the contract. This can be done if no existing Service covers the needs. But still it is necessary to make the Service general enough to be used in other scenarios [18, p. 8]. • Mandated - when the provider singly define the contract. This can also be seen as a “take it or leave it” approach [18, p. 8]. <p>In order to support the Service Life Cycle management it is necessary to tightly control the Partner Integration Policy.</p>
Potential derived artifact(s)	<p>Outsourcing strategy SOA adds a new level of abstraction on IT enabling easier outsourcing of processes [106]. However, this must be aligned with an overall strategy.</p>
Relation to EA	<p>Integrating with other companies is indeed a strategic decision. Using SOA this can be incorporated directly into the SOA Adoption Road Map.</p>

Table 17 : SOA Artifact # 11

Title	Quality control
Description	<p>Quality control is by no means a special case for SOA, however the importance of quality control becomes perhaps even more critical in the context of SOA. In order to control the evolution of an SOA, the quality of the implemented Services must be defined in a unanimous way, and kept under constant monitoring. The organisation must define a quality assurance process for Services, which the Services must pass before they are published for public use.</p> <p>The issue of SOA is when publishing a Service, the whole purpose is that the Service is to be used in different contexts than for which it is developed - which gives several challenges. An example is if a new Service-consumer has higher requirements to Quality of Service (QoS) than a given Service can provide. If these requirements are not related to the functionality provided by the Service, such as lower guaranteed response time, the Service will need an update. Implementing a parallel Service providing the same functionality, only diverted by a different Service Level Agreements (SLA), will violate the very foundation of SOA. In SOA the Service should be updated to ensure that the benefits are shared.</p>
Potential derived artifact(s)	<p>Service Level Agreements A Service Level Agreement (SLA) is a formal contract between a Service provider and a Service consumer - which can be used in both an internal-</p>



and external context. Often the SLA is only seen as being a matter of technical demands, such as 99.9% uptime, but this is only one side of the story. Non technical issues should also be covered by the SLA, such as contact information to the provider and legal issues [108].

Agreeing on a SLA between two Services is one thing, but what if the demands to a Service changes and the SLA must be changed? In a SOA context a Service is often created as a composite of other Services for which there are individual SLA's, and these can then also be subject to change-requests. In order to avoid a chaos of negotiating SLA's, it is important to have a standard method of creating SLA's supported by a department with the proper knowledge and authority.

Version Control (change control)

The nightmares of versioning have always been an issue in IT, and do not stop at SOA. In fact, I often talk with people that see SOA as making the nightmare even worse. The argument is often, that SOA's ability to change, in order to support agility, will result in a mayhem of Services in different versions. The problem is that you cannot just change a Service as this can potentially make the entire IT system shut down.

There are arguments enough implying SOA will result in a versioning nightmare, but I would like to turn this argument around; SOA will not magically remove the challenges of versioning, but it will however give you opportunity to control the versioning at a central unit and thereby apply a common strategy of versioning.

Reuse policies [46]

Deploying Services and reusing these in new contexts is essentially how SOA works. But in order to make this idea feasible a Reuse Policy must be formulated [46]. Although SOA is often illustrated by a Service Consumer looking up the needed functionality in a registry and thereafter connecting to the relevant Service Provider, I see this as a very simplified model. This simple model can be used in some cases but is essentially just an advanced Lookup-Service and must not be confused with a Reuse Policy. A Reuse Policy must cover how reuse is done in a strategic sense, dividing cost etc.

Service Documentation rules

A Service without documentation will have little or no possibility of being reused. The Service is not fully defined by its WSDL⁴⁵, the WSDL might define the methods and data of the Service, but in order to describe a Service much more information is required. A Service should support a business process that should be documented as well. There are many more issues that need documented in order to make a Service reusable, but it is most important to have a standard way of doing it.

Security models [46]

Security is often pointed out as one of the drawbacks of SOA, with the argument that SOA opens up your systems, making them more susceptible to security breaches. This is an argument that holds some truth, but is also an area where a lot of effort is being put to develop standards to ensure the security. However the work being done, is mainly

⁴⁵ The Web Services Description Language (WSDL) is a format for describing Web Services in XML



	<p>focused on the technical issues, opening up your systems has many other security aspects; an example is who should have access to what information? This is as such not a special case for SOA, but SOA makes the issue very clear and a strategy must be formed to ensure a uniform approach to security issues across the entire SOA.</p> <p>Test procedures Testing of software is an old and well established science which is also relevant for SOA. The issue that makes it even more important in a SOA context is that one faulty Service can shut down the entire system. A methodology must be formed to ensure all Services deployed are properly tested.</p>
Relation to EA	Quality takes time, and time is money. Quality is here discussed as a broad spectrum of issues that are primarily concerning the development of policies. Policies of such general importance that they must be anchored high in the organisation; at EA level.

Table 18 : SOA Artifact # 12

Title	Develop SOAD
Description	<p>Often the notion is that developing Services is just “developing as usual” just using the Web Service standards. This is a very dangerous path to take. By doing this you devalue SOA to just being a technical issue. Developing Services for an SOA is much more complex as the Services must be implemented to support actual business processes [38].</p> <p>Since no software development methods presently support the actual development Services in an SOA [26], there lies a great challenge in defining a method. A challenge that could look like the re-emergence of the Method Department with tasks like; collecting, evaluating and sharing experiences from projects.</p>
Potential derived artifact(s)	<p>SOA method staff A staff to advice and guide new projects should be formed. They can be the stakeholders of “compliance control”.</p>
Relation to EA	The development method can be seen as the tool to implement EA into IT. This is not a special case for SOA but a general issue.

Table 18 : SOA Artifact # 13

Title	Enterprise Service Bus Policy
<p>Description</p>	<p>There are generally two approaches on “plumbing” SOA; Peer to Peer or using an Enterprise Service Bus (ESB):</p> <p>Peer to Peer, will usually include a discovery service, such as an UDDI, but after the discovery the Services communicate directly one-to-one.</p> <p style="text-align: center;">Figure 27: SOA peer to peer [Source: 31]</p> <p>The peer to peer approach works as follows; Web Service A makes a lookup in the Service Registry, gets the location of Web Service B, and the communication between A and B can commence.</p> <p>This view of SOA will not provide the long term agility that is the purpose of SOA. More is needed to ensure that the SOA can evolve over time, and include automation of some of the governance issues, such as SLA monitoring etc.</p> <p>Enterprise Service Bus⁴⁶ (ESB) has become a widely accepted concept. In 1999 David Sprott introduced the concept of Business Service Bus [11, p. 5]. Since 1999 the name Business Service Bus has changed to Enterprise Service Bus⁴⁷, as well as the ideas of David Sprott, was somewhat academic in 1999, today they are almost available from all major vendors on the market [10, p. 16].</p> <p>The reason I see the choice, between whether to use an ESB or peer to peer, as a product from the SOA-process is somewhat aligned to how I see SOA.</p>

⁴⁶ In section 9.2 I referred to this concept as; SOA Application Architecture. Gartner refers to this concept as the Enterprise Nervous System [76].

⁴⁷ The concept of a Enterprise Business Bus still exist, but is usually seen as a subset of the ESB.

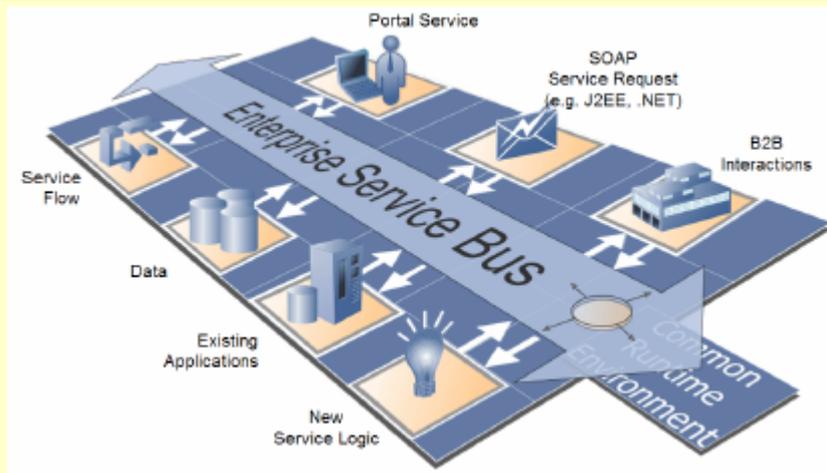


Figure 28: The Enterprise Service Bus [Source: 2]

The definition of SOA often includes an Enterprise Service Bus (ESB), as illustrated in Figure 28. It is my belief that in order to implement a full scale SOA, some type of ESB must be implemented [2, p. 74] [30]. However, an ESB is not as much a product as it is a concept, although many vendors proclaim to have an ESB on their shelf⁴⁸ [111]. The definition of which responsibilities the ESB shall implement is the product - I will refer to this product as the "ESB-policy".

Potential derived artifact(s)

ESB Capability Model

A model to specify capabilities is the responsibility of the ESB. The content of this model is of course dependent on definition of the ESB-policy, and an example of an ESB Capability Model could look like:

Communication	Service interaction
<ul style="list-style-type: none"> • Routing • Addressing • Protocols and standards (HTTP, HTTPS) • Publish / subscribe • Response / request • Fire & forget, events • Synchronous and asynchronous messaging 	<ul style="list-style-type: none"> • Service interface definition (WSDL) • Substitution of service implementation • Service messaging models required for communication and integration • (SOAP, XML, or proprietary Enterprise • Application Integration models) • Service directory and discovery
Integration	Quality of service
<ul style="list-style-type: none"> • Database • Legacy and application adapters • Connectivity to enterprise application integration middleware • Service mapping • Protocol transformation • Data enrichment • Application server 	<ul style="list-style-type: none"> • Transactions (atomic transactions, compensation, WS-Transaction) • Various assured delivery paradigms (WS-Reliable Messaging or support for Enterprise Application Integration middleware)

⁴⁸ As Commercial Of The Shelf (COTS).



	environments (J2EE and .Net)	
	<ul style="list-style-type: none"> Language interfaces for service invocation (Java, C/C++/C#) 	
	Security	Service level
	<ul style="list-style-type: none"> Authentication Authorization Non-repudiation Confidentiality Security standards (Kerberos, WS-Security) 	<ul style="list-style-type: none"> Performance Throughput Availability Other continuous measures that might form the basis of contracts or Agreements
	Message processing	Management and autonomic
<ul style="list-style-type: none"> Encoded logic Content-based logic Message and data transformations Message / service aggregation and correlation Validation Intermediaries Object identity mapping Service / message aggregation Store and forward 	<ul style="list-style-type: none"> Administration capability Service provisioning and registration Logging Metering Monitoring Integration to systems management and administration tooling Self-monitoring and self-management 	
Modelling	Infrastructure Intelligence	
<ul style="list-style-type: none"> Object modelling Common business object models Data format libraries Public versus private models for business-to-business integration Development and deployment tooling 	<ul style="list-style-type: none"> Business rules Policy-driven behaviour, particularly for service level, security and quality of service capabilities (WS-Policy) Pattern recognition 	
<p>Table 19 : Categorized Enterprise Service Bus capabilities [Source: 2, p 83]</p> <p>A central issue when identifying the desired ESB capabilities is to identify which of the capabilities that can be implemented as an automated process in the ESB!</p>		
Relation to EA	The ESB policy is the definition of how the Nervous System of IT works inside the enterprise, hence all new projects must oblige to the ESB.	

Table 20 : SOA Artifact #14

Title	Use of standards
Description	SOA is based on the philosophy of the use of standards. However, it is necessary to have mechanisms that evaluate if a standard is applicable. Standards are in them selves no guarantee for ability to create interoperable systems. This was also described in section 7.3.
Potential derived artifact(s)	Authority Model Selecting a standard this does not necessarily mean that it will be



	<p>practically applicable in all contexts. I attended a conference where A.P. Moller Maersk Group⁴⁹ explained their authority model regarding use of standards. The model divides the standards into four categories:</p> <ul style="list-style-type: none"> • Mandatory - all projects must oblige • Recommendation - reasons not to use the standard must be documented. • Directional - The standards have been evaluated and are approved. • Informational - These standards can be relevant. <p>Another approach is that of the Danish government: The Interoperability Framework. Here the following categories are [123]:</p> <ul style="list-style-type: none"> • Recommended: A recommended standard is crucial to the interoperability of an enterprise (system) and should be enforced. For new development all recommended standards should be carefully considered. • Approved: Approved standards have generally proved their value and are considered to be mature. • Emerging: Emerging standards may have future value within the enterprise but have proven no specific benefit at the time. The enterprise may be conducting a pilot project to establish the potential benefits and risks of selecting this standard. • De Facto: A De Facto standard identifies choices that are accepted because of widespread use within the industry. • Sustained: A Sustained standard indicates a standard or practice that no longer shows promise but is still used or even expanded because of a prior standard solution. • Migrate From: A Migrate From designation refers to a standard or practice that has been abandoned for a better solution. It is not a favoured standard yet continues to be in use in some enterprises. Enterprises should plan to migrate away from solutions assigned with this designation as soon as practical. <p>The key difference between that of the Danish government and A.P. Moller Maersk Group is the Mandatory category of A.P. Moller Maersk Group. This is not included in The Interoperability Framework of the Danish government. However there exist another possibility in the governmental framework; a standard can become mandatory by law. An example of this is one of the 1st of February 2005 where all invoices are to be sent electronically using an adapted version of the Universal Business Language UBL⁵⁰.</p>
Relation to EA	<p>The two examples of authority models are both examples of decisions made in the context of EA programs. What is important to recognize is that the decisions can have immense implications on the SOA. Standards must be evaluated at an enterprise-wide level in order to avoid conflicting standards.</p>

Table 21 : SOA Artifact # 15

⁴⁹ <http://www.maersk.com>

⁵⁰ http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl



All in all I identified 15 high level SOA Artifacts and 23 derived Artifacts, giving a total of 38 Artifacts, and I will not identify any further SOA Artifacts. The purpose here is not to create the complete list of SOA Artifacts - this is simply impossible. The number and type of SOA Artifacts are depended on the given scenario. This is a context free scenario with the purpose to identify issues where SOA Artifacts will change EA.

As stated, there are two ways the Artifacts of SOA will affect EA:

1. It can change EA.
2. It can add new aspects.

Relating these two aspects to how SOA Artifacts will change EA Artifacts is quite a mind-puzzling exercise, and a parallel to the discussion in section 9 on the joint evolution of SOA and EA. This exercise can be made on the Artifact level as well:

1. EA Artifacts are absorbed by SOA Artifacts
2. SOA Artifacts are absorbed by EA Artifacts
3. EA- and SOA Artifacts will merge into a new concept

This issue is dependent on the result from the same exercise on SOA and EA in general (see section 9), where I stated that SOA and EA would merge into the concept SOEA, this merging-process I also believe is what will happen with regards to SOA- and EA Artifacts.

9.2.1 Types of SOA Artifacts

I have in the previous section focused on describing the SOA Artifacts and their affect on EA, but I have also stated that the SOA Artifacts can be divided in two categories. I will in the following table list the identified high level SOA Artifacts and mark their category:

- **Pure SOA Artifact** - are Artifacts that has emerged with the introduction of SOA.
- **Modifies EA Artifact** - are Artifacts that essentially should be part of any EA, but will be modified by the introduction of SOA.

The placement in the two categories is based on the notion that there is an existing EA program.

Artifact #	Artifact name	Pure SOA Artifact	Modifies EA Artifact
1	Definition of SOA team IF an EA team existed a new team should not be created, however new qualifications must be added to the team.		X
2	Defining SOA concepts Defining concepts for the enterprise is an EA task.		X
3	Documentation Framework I have placed this as modifying EA to is to ensure a strong coupling to EA.		X
4	Defining the goals for SOA (SOA Vision) SOA is such a big endeavour that it should be part of the overall strategy of the business.		X
5	Making the business think SOA EA is strongly founded in the business. One of the responsibilities of an EA program is to communicate strategies and concepts.		X
6	SOA adoption roadmap		X



	As discussed in section 6.1.2 the EA process is about getting from the current state to the target state, and SOA being the target state makes the SOA Adoption Roadmap the foundation for the EA process.		
7	Return of Investment Plan.		X
	Specifying the desired ROI is a central issue of EA, this must be aligned with SOA as the timescale of the ROI has a big impact on SOA.		
8	Pricing Policy Model (PPM)		X
	"Who is paying" is a question often asked, and the answer should be identified as a part of EA. However in SOA the answer critical for the success of SOA.		
9	Data Model Policy		X
	Data has a dedicated column in the Zachman Framework, however SOA will change how data is perceived as a new layer is introduced (see section 8.3)		
10	SOA Governance		X
	EA without governance makes little sense. The aspects of SOA will have a great impact on how to govern not only IT, but the business as well.		
11	Partner Integration Policy		X
	SOA is not a prerequisite for having a Partner Integration Policy, but SOA can be the tool to improve issues of outsourcing and partner integration.		
12	Quality control		X
	Quality control is probably one of the oldest disciplines of IT. SOA will not fundamentally change this, the important issue when talking quality in an SOA context is however that it must be seen as an enterprise-wide issue.		
13	Develop SOAD		X
	There are many software development methods throughout an enterprise. In an SOA context they must be seen as an enterprise-wide issue.		
14	Enterprise Service Bus Policy	X	X
	The ESB-policy will affect all IT projects, and any decision which sets out rules for the enterprise must essentially be founded in EA. The concept of an ESB is closely tied to SOA - hence I have placed this SOA product in both categories.		
15	Use of standards		X
	Use of standards must be mandated at an enterprise level to ensure non-conflicting standards. This is not a special for SOA, but as SOA is based on the use of standards SOA will have great influence on this issue.		

Table 22 : Categorized high level SOA Artifacts

I stated that there would only exist two categories of SOA Artifacts, however all the identified high level SOA Artifacts are placed in the same category; the "Modifies EA Artifact". This of course raises the question if SOA will only modify the discipline of EA. Another possibility is that the issues concerning SOA have risen to a level of abstraction similar to EA. In section 7.1 I discussed the fact that SOA had evolved from a "bottom-up"- to a "top-down" paradigm. This is in my view the reason for the findings illustrated by Table 22. Not only has the level of abstraction of SOA reached that of EA, but also the perspective of which both are approached are now equal.

This is indeed a very interesting issue on which I will return, but in order to complete the picture I will in the following continue the work of Table 22, and include the derived SOA Artifacts.



Artifact #	Artifact name	Pure Artifact	SOA	Modifies Artifact	EA
1	Definition of SOA team			X	
1.a	<i>Definition of authority</i>			O	
	As the SOA team already exists in the form of an EA team it must be assured that the proper authority has been given to the team. However with the introduction of SOA there might be new areas that will change this issue - which could mean both adding new areas of authority, but could also mean removing areas of authority from the team.				
2	Defining SOA concepts			X	
2.a	<i>Reference Model</i>		O		
	The reference model of SOA is the template of SOA within the enterprise.				
3	Documentation Framework			X	
3.a	<i>Service map</i>		O		
	Is a new product founded in SOA.				
4	Defining the goals for SOA (SOA Vision)			X	
4.a	<i>Scoping</i>			O	
	The issue of scoping must be seen in a bigger context than SOA itself.				
5	Making the business think SOA			X	
5.a	<i>Education</i>			O	
	Education is a part of the responsibilities grounded in EA.				
6	SOA adoption roadmap			X	
6.a	<i>"As is" analysis</i>			O	
	This is an essential part of EA (section 6.1.2)				
6.b	<i>"To be" analysis</i>			O	
	This is an essential part of EA (section 6.1.2)				
7	Return of Investment Plan.			X	
7.a	<i>Specify ROI</i>		O		
	Making the SOA measurable will be an effort closely related to how the SOA of the enterprise is defined.				
8	Pricing Policy Model (PPM)			X	
8.a	<i>PPM standard</i>		O		
	In order to manage the PPM automatically this will be closely integrated to the SOA is to be implemented.				
8.b	<i>Setup PPM staff</i>			O	
	I do not see staffing issues related to SOA as the authority must be defined at enterprise level.				
9	Data Model Policy			X	
9.a	<i>Identifying data owners</i>		O	O	
	This is mainly a political issue which should have been clear in any enterprise, but is a necessity of SOA, and should be seen in a new light as a result of SOA - hence placed this in both categories.				
9.b	<i>Data Model Staff</i>			O	
	I do not see staffing issues related to SOA as the authority must be defined at enterprise level.				
9.c	<i>SOA Data Model</i>		O		
	Is a new product founded in SOA.				
10	SOA Governance			X	



10.a	<i>Service choreography standard</i>	O	
	Is a new product founded in SOA.		
10.b	<i>Service Ontology</i>	O	
	Is a new product founded in SOA.		
11	Partner Integration Policy		X
11.a	<i>Outsourcing Strategy</i>		O
	SOA is the tool of how it can be done.		
12	Quality control		X
12.a	<i>Service Level Agreements</i>	O	O
	The concept should be a part of EA, but new requirements arise with SOA.		
12.b	<i>Version Control (change control)</i>	O	O
	The concept should be a part of EA, but new requirements arise with SOA.		
12.c	<i>Reuse policies</i>	O	O
	The concept should be a part of EA, but new requirements arise with SOA.		
12.d	<i>Service Documentation Rules</i>	O	O
	Documentation rules should be defined as a part of any EA, but new requirements arise with SOA.		
12.e	<i>Security Models</i>	O	O
	The concept should be a part of EA, but new requirements arise with SOA.		
12.f	<i>Test Procedures</i>	O	O
	The concept should be a part of EA, but new requirements arise with SOA.		
13	Develop SOAD		X
13.a	<i>SOA Method Staff</i>		O
	I do not see staffing issues related to SOA as the authority must be defined at enterprise level.		
14	Enterprise Service Bus Policy	X	X
14.a	<i>ESB Capability Model</i>	O	
	SOA makes this possible.		
15	Use of standards		X
15.a	<i>Authority Model</i>		O
	This is a general part of EA.		

Table 23 : Categorized derived SOA Artifacts

The purpose of including the derived SOA Artifacts into the table of categorised Artifacts was to identify if any of the derived SOA Artifacts would be in the category of “pure SOA Artifacts”. The notion was that the derived Artifacts would have a lower abstraction level, hence separating the relation to EA. Table 23 shows indication of this notion being correct, as fourteen of the derived Artifacts in fact are placed as pure SOA Artifacts.

The placing of the Artifacts is not black and white and is depended on the given context and the level of maturity of SOA and EA (as discussed in section 7.5). To discuss in depth why each of the SOA Artifacts are placed in the given category is not possible in the scope of this thesis, this could be the basis of a complete thesis. I will however select a few of the Artifacts and go into greater depth with these, but before commencing this I will look at the relation between EA- and SOA Artifacts.



9.3 SOA Artifacts vs. EA Artifacts

The concept of an Artifact is, as discussed in section 9.1, a very broad concept. Most literature is focusing on the definition of what EA Artifacts is, and I have had little luck finding examples of actual EA Artifacts. However, I have had the great privilege to get a preview of the second edition of Scott A. Bernard's: "An Introduction to Enterprise Architecture". In this edition he is not only talking about what an EA Artifact is, but is also listing his view on recommended EA Artifacts, and maps these into the Zachman Framework.

Getting this new material gives the opportunity to couple my SOA Artifacts with EA Artifacts. I will not describe the Artifacts identified by Scott A. Bernard, but in order to show the nature of the EA Artifacts I will list these in the following table. Furthermore I will use these EA Artifacts to identify the relations with the SOA Artifacts that I have identified.



EA3 Cube Level/Thread	Artifact ID #	Artifact Name (* Composite Artifact)	Zachman Mapping	Modified by SOA Artifact(s)
Strategic Goals & Initiatives (I)	S-1	Strategic Plan*	C6/R1	4 Defining the goals for SOA (SOA
	S-2	SWOT Analysis	C5/R1	4.a Scoping
	S-3	Concept of Operations		4.a Scoping
	S-4	Concept of Operations	C2/R1	4.a Scoping
	S-5	Balanced Scorecard™ *	C6/R4, C6/R5	4.a Scoping
Business Products & Services (B)	B-1	Business Plan*	C2/R2, C5R1	7. Return of Investment Plan.
	B-2	Node Connectivity Diagram	C3/R1	3.a Service map
	B-3	Swim Lane Process Diagram	C4/R2	1.a Definition of authority
	B-4	Business Process/Service Model	C2/R2	3.a Service map, 9.a Identifying data owners
	B-5	Business Process/ Product	C4/R2	
	B-6	Use Case Narrative &	C6/R3, C6/R4	
	B-7	Investment Business Case*		7.a Specify ROI
Data & Information (D)	D-1	Knowledge Management Plan	C1/R1, C1/R2	9 Data Model Policy
	D-2	Information Exchange Matrix*	C3/R2, C4/R2	9.a Identifying data owners
	D-3	Object State-Transition	C1/R3	
	D-4	Object Event Sequence	C2/R2, C5/R3	14 Enterprise Service Bus Policy
	D-5	Logical Data Model	C1/R3	9.c SOA Data Model
	D-6	Physical Data Model	C1/R4	9.c SOA Data Model
	D-7	Activity/Entity (CRUD) Matrix *	C1/R3, C4/R2	Not relevant in SOA (loosely
	D-8	Data Dictionary / Object	C1/R5	9.c SOA Data Model
Systems & Applications (SA)	SA-1	System Interface Diagram	C3/R4, C3R2	3.a Service map
	SA-2	System Communication	C2/R4, C3/R2	14.a ESB Capability Model
	SA-3	System Interface Matrix *	C2/R4	3.a Service map
	SA-4	System Data Flow Diagram	C2/R3	3.a Service map
	SA-5	System/Operations Matrix *	C2/R4	3.a Service map
	SA-6	Systems Data Exchange	C2/R3	3.a Service map
	SA-7	System Performance Matrix *	C2/R3	12.a Service Level Agreements
	SA-8	System Evolution Diagram	C2/R4	6 SOA adoption roadmap, 10 SOA Governance
	SA-9	Web Application Diagram	C2/R3	14 Enterprise Service Bus
Networks & Infrastructure (NI)	NI-1	Network Connectivity Diagram	C3/R5	
	NI-2	Network Inventory	C3/R5	
	NI-3	Capital Equipment Inventory	C3/R5	
	NI-4	Building Blueprints *	C3/R5	
	NI-5	Network Center Diagram	C3/R5	
	NI-6	Cable Plant Diagram	C3/R5	
	NI-7	Rack Elevation Diagram	C3/R5	
Security (SP)	SP-1	Security and Privacy Plan*	C4/R5	12.e Security models
	SP-2	Security Solutions Description	C4/R5	12.e Security models
	SP-3	System Accreditation	C4/R5	12 Quality control
	SP-4	Continuity Of Operations Plan*	C4/R5	
	SP-5	Disaster Recovery Procedures	C4/R5	
Standards (ST)	ST-1	Technical Standards Profile	C3/R4	15 Use of standards
	ST-2	Technology Forecast	C3/R4	
Workforce (W)	W-1	Workforce Plan*	C4/R1	
	W-2	Organization Chart	C4/R2	
	W-3	Knowledge and Skills Profile	C4/R3	

Table 24 : Enterprise Architecture Artifacts [Source: 110]

As the relations between EA and SOA have already been established this exercise is only to proof of this further. The interesting issue here is relating Table 24 to Table 22 and Table 23. Table 22 and Table 23 categorised the SOA Artifacts into; “Pure SOA Artifact” and “Modifies EA Artifact”. The natural consequence to this is that only the SOA Artifacts that are in the “Modifies” category should appear in Table 24 - hence, if they are categorised as “Pure SOA

Artifacts” they should not be related to any existing EA Artifact. All the SOA Artifacts categorised as “Modifies EA Artifact” should be mapped into Table 24. However, this is not the case with neither of them.

The reason that not all the SOA Artifacts, categorised as modifying EA Artifacts, are mapped into Table 24 is because it is not a complete list. It is only a set of what Scott A. Bernard recommends as the most important. However that there are several “Pure SOA Artifacts” mapped against the EA Artifacts is more interesting. Essentially this can only be interpreted as; either is the SOA Artifacts categorised incorrect, or the EA Artifacts listed by Scott A. Bernard are already influenced by the thoughts of SOA! The Pure SOA Artifacts mapped into Table 24 are:

- 3.a Service Map
- 9.a Identifying data owners
- 9.c SOA Data Model
- 14 ESB policy
- 14.a ESB Capability Model

The Service Map is represented seven times as related to EA Artifacts, which is strong indication that this is definitely an SOA Artifact affecting EA. Is it therefore categorised incorrectly in Table 22 and Table 23? That the SOA Artifacts are categorised as “Pure SOA Artifacts” is done from the view that they are mainly concerned with issues only relevant in a SOA context. This is the case for the Service Map. The Service Map is one of the main governance tools in SOA, but will be relevant at many levels of abstraction. This is the reason for the many relations to the EA Artifacts.

Looking at the following two SOA Artifacts in the list above they both concern how data is seen within the enterprise. I have discussed this issue with regards to SOA in section 8.3. There is no doubt that this is one of the issues where SOA has a major impact on EA, but the impact is of such great dimensions that I do not see it as change, but as a new way of working with data throughout the enterprise.

The last two in the list concern the IT infrastructure. Here the EA Artifact defined by Scott A. Bernard is identical to the SOA Artifacts. The EA Artifact: SA-9 Web Application Diagram (shown in Figure 29) is in my view identical to that of an ESB defined in SOA Artifact # 14.

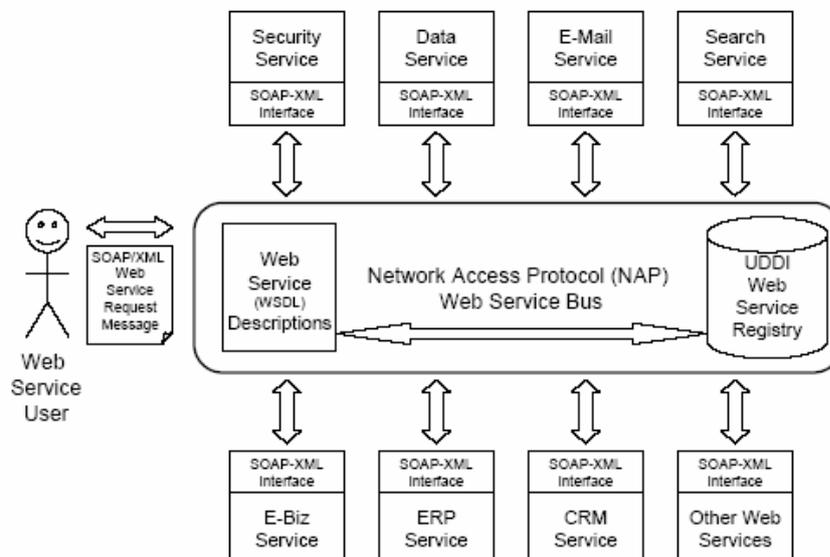


Figure 29 : SA-9: Web Application Diagram [Source: 110]



This is a very interesting evolution as this shows that SOA Artifacts are already seen as EA Artifacts! Essentially this alone can be seen as proof of SOA is affecting EA. However, I see this as an unfortunate development as this is done without looking at the consequences.

Before I go into further depth with the SOA Artifacts, and how they relate to EA, I will look at the relations between the SOA Artifacts themselves. My motivation for this thesis was to show that in order to get the promised agility of SOA there is not only a need to connect SOA closely to EA, but it will also be of the utmost importance to understand how to handle the challenges of an ever changing enterprise. In order to cope with change it is necessary to know the consequences of change; what related aspects will be affected as a consequence of the given change.

9.4 SOA Artifact relations

As described in section 6.1.1.1 the motivation behind EA is the ability to cope with change with a minimum of effort. In complex structures, change in one place will create change in other places. The ability to adapt to change is therefore not just adapting in one place but can mean change in many places - change as a result of change. The difficult issue is to know where to change!

The first step in trying to prepare one self for change is to identify the relation between aspects that can change, and as everything potentially can change it is of great importance to scope the changeable aspects (as in SOA Artifact # 4.a Scope).

Looking at how change will affect SOA it is important to know about the relations between the SOA Artifacts as:

- An Artifact which is not updated will loose its value.
- Updating an Artifact will often be derived by a change in another Artifact.

This rippling effect of change entails complexity and is something that needs a high level control and governance. In section 5 on hype of SOA one of the hypes was that SOA would make Governance easier, which seems rather questionable. It seems that SOA does not solve any complexity problems. However, a lot of the complexity can be removed, if one is aware of the challenges of SOA, and have developed methods to control the evolution of SOA.

I will in the following look further at the SOA Artifacts identified in Table 22. I will only look at the high level Artifacts as the purpose is not to define the complete set of relations, but to illustrate the complex pattern of relations between the Artifacts. Further more the relational map can be seen as an Artifact itself - a sub-Artifact of SOA Artifact # 3 Documentation Framework.

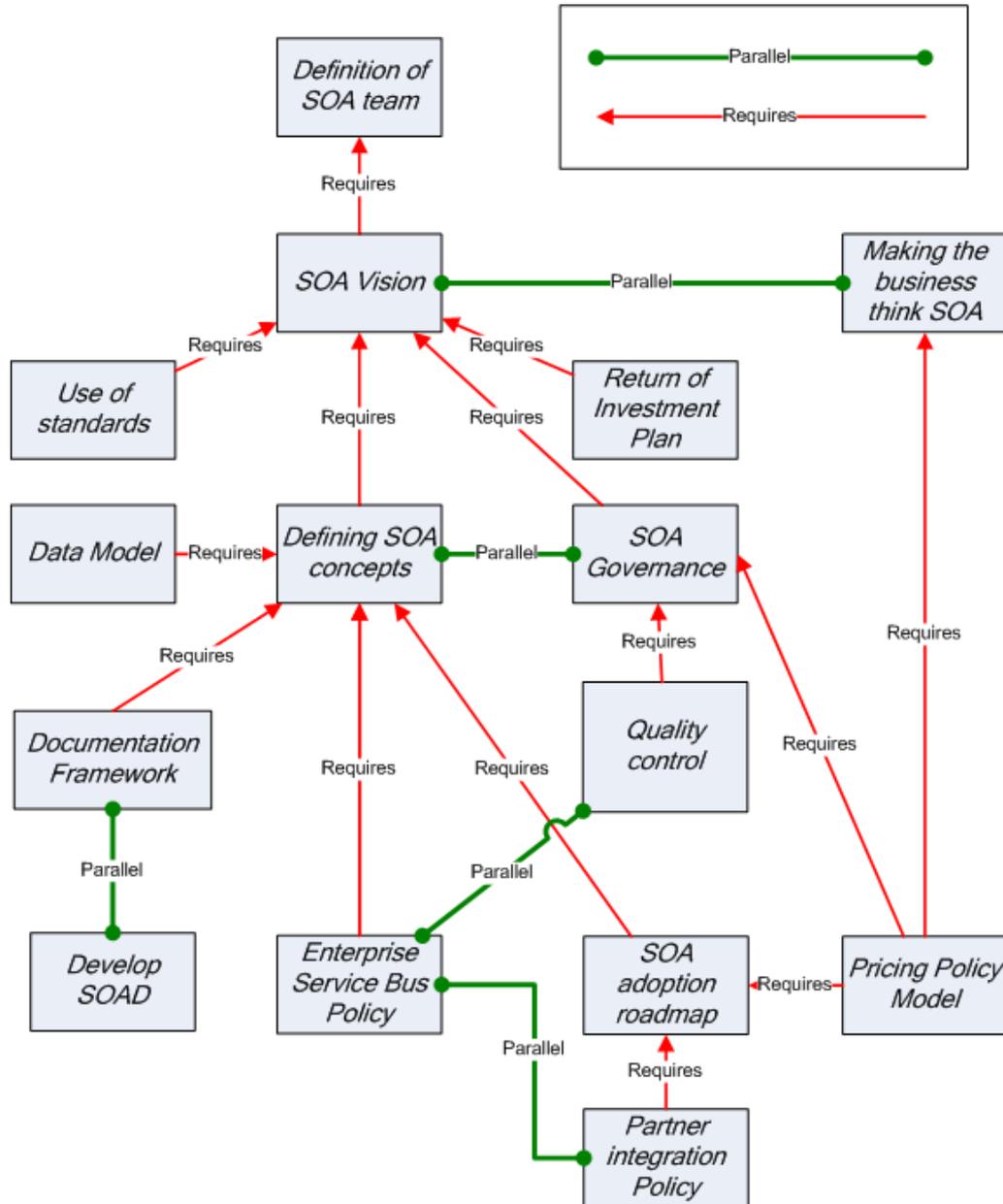


Figure 30 : Relations between SOA Artifacts [Source: own work]

Figure 30 consists of three elements:

- The high level SOA Artifacts
- Red arrows to illustrate that an Artifact requires the existence of another Artifact.
- Green connectors to indicate Artifacts that must be defined in a parallel process.

Identifying the relations between the Artifacts is not an easy task, and essentially one can argue that all the Artifacts are related. The focus in Figure 30 is not to identify all the relations, but to give a picture of the complexity that exists when trying to identify these. As noted earlier one can not entail all possible changes in a general model - every change will have different consequences depending on the given context. I will use the model in Figure 30 as a frame of reference in order to further elaborate on two selected SOA Artifacts.



9.5 In-depth analysis of two SOA Artifacts

Agility is the prime goal of SOA, but agility and change are like yin and yang. This has formed the basis for my selection of the two SOA Artifacts which I will further analyse:

- # 14 ESB Policy
- # 12.b Version Control

The reasoning for the choice of these is that the ESB Policy is a great part of making the agility practically possible. Change will show the need of Version Control and is also identified as a SOA Artifact derived from Quality Control.

9.5.1 Enterprise Service Bus Policy

One of the arguments I often meet when discussing the practical implementation of an Enterprise Service Bus (ESB) is that it is a dangerous approach giving a single point of failure. This is often be true, but can also be seen as a big part of the motivation for having an ESB as it will be possible to focus on making the ESB the most reliable part of the system. But bare in mind that an ESB is, like SOA, not a “silver bullet” that will solve all your problems [111].

The issues of seeing the ESB being a single point of failure can be seen in parallel to the differences between the ESB Policy and the ESB implementation. As many vendors claim to have an ESB ready for purchase they evidently also claim that they have a definition for what an ESB is [113].

Although I have not yet defined exactly what ESB is I have already been discussing it as though it is a well defined concept. However like so many other concepts in the world of SOA this is far from the truth. To illustrate this I will give you a couple of different definitions [113]:

“A Web-services-capable infrastructure that supports intelligently directed communication and mediated relationships among loosely coupled and decoupled biz components.”

[Gartner Group]

“The ESB label simply implies that a product is some type of integration middleware product that supports both MOM and Web services protocols.”

[Burton Group]

“A standards-based integration backbone, combining messaging, Web services, transformation, and intelligent routing.”

[Sonic Software]

“An enterprise platform that implements standardized interfaces for communication, connectivity, transformation, and security.”

[Fiorano Software]

“To put it bluntly: If you have WebSphere MQ and other WebSphere brokers and integration servers, you have an ESB.”

[Bob Sutor, IBM]

“The Enterprise Service Bus is a uniform service integration architecture of infrastructure services that provides consistent support to business services across a defined ecosystem. The ESB is implemented as a service oriented architecture using Web Service interfaces.”

[CBDI]

What is an ESB? A question with no single answer, but with many opinions. An interesting point here is that IBM has just released⁵¹ their WebSphere Enterprise Service Bus⁵². So, is Bob Sutor right in his statement cited above? I will leave this as an open question, because the important issue in this context is where the ESB fits within SOA.

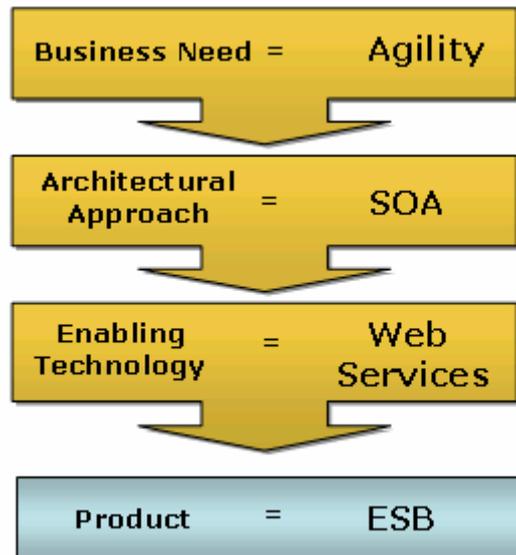


Figure 31 : From business needs to ESB [Source: 112]

Figure 31 is an example of a vendors⁵³ view on where the ESB fits in the big picture. Their view on the ESB being the product itself is clearly motivated by their desire to sell a product. However putting this aside they do have the view that the ESB is the practical implementation that can support the agility needs of the business. I will not dwell further in the definition of the ESB - it is essentially a matter of definition for each enterprise!

Since such a definition does not exist, it is of the utmost importance that you define what an ESB is in the context the given enterprise (SOA Artifact #14: ESB policy). This will enable you to select between the different vendors, the product that best fits your needs.

The purpose of this section is to look at how the ESB Policy relates to the other identified SOA Artifacts using the relations illustrated in Figure 30. The following illustration is extracted from this focusing on the Enterprise Service Bus Policy Artifact.

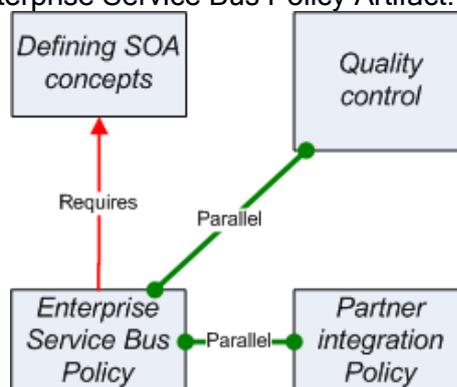


Figure 32: Relations to ESB Policy [Source: own work]

⁵¹ Being released September 05

⁵² <http://www-306.ibm.com/software/integration/wsesb/>

⁵³ Cape Clear



Looking at the relations of the ESB Policy it seems somewhat simple, however it must be bared in mind that changes in other SOA Artifacts will ripple through the SOA Artifacts on which adding many hidden relations, and furthermore it is only the high level SOA Artifacts that are included in the model.

I will in the following look at how the SOA Artifacts depicted in Figure 32 are related to the ESB Policy.

Defining SOA Concepts

The ESB Policy requires that the concepts of SOA are defined. As discussed the concept of an ESB is not known and could have been placed as a part of the SOA Artifact # 1 : Defining SOA Concepts, however I see the ESB Policy as such a central issue of SOA that it must be isolated as an independent SOA Artifact. However making a separate SOA Artifact does not remove the strong dependency to how the other concepts of the SOA world are perceived.

The strong connection between the ESB and other SOA concepts are often directly related as many of the general SOA concepts will be practically implemented in the ESB. A great deal of the ESB Policy will be to define which should be covered by the ESB.

Quality Control

As discussed, Quality Control is of the utmost importance in an SOA. The development of the ESB Policy is marked as a parallel to the Quality Control Artifact. This is done from the notion that many aspects of Quality Control in an SOA must be done automatically in order to make it practically feasible. Usually control within an ESB context means some kind of monitoring. An example of this is monitoring if Services comply with their SLA. Not all parts of an SLA can be monitored automatically and this must be thought into the design of how to make ones SLA - there is little sense in making SLA's that can't be controlled. The design of the SLA's must be aligned with the capabilities of the ESB.

Partner Integration Policy

Again here is a parallel Artifact. Making a policy on how business partners must integrate with the system of the Enterprise is of course a sensitive issue. But, as with the quality control you can not integrate in ways that are not supported by the ESB. Therefore you must align the ESB Policy with the wishes of Partner Integration Policy, and vice versa.

Missing relations

Even though many of the relations reach the ESB Policy through the “rippling effect”, there are several SOA Artifacts that does not “connect” with the ESB Policy even though a connection could seem obvious. An example of this is that there is no direct or indirect relation to the Pricing Policy Model (PPM), even though there would, as with Quality Control, be a need for automation. As stated earlier there is implicitly a connection between all the SOA Artifacts.

Mapping to Zachman Framework

I have mapped the identified SOA Artifacts into the work of Scott A. Bernard (see Table 24) and he has mapped his EA Artifacts into the Zachman Framework. This makes it possible to map the ESB Policy into the Zachman Framework. The following illustration is a minimized version of the Zachman Framework with the EA Artifacts that are related to the ESB Policy plotted in using their reference id from Table 24. The ESB Policy is related to; D-4 Object Event Sequence Diagram and SA-9 Web Application Diagram (See Appendix C).

	What	How	Where	Who	When	Why
Scope {Contextual}						
Business Model {Conceptual}		D-4				
System model {Logical}		SA-9			D-4	
Technology model {Physical}						
Detailed rep. {Out of context}						

Figure 33: EA D-4 and SA-9 mapping [Source: own work]

Scott A. Bernard does not argue for his placing of his EA Artifacts, therefore I will shortly discuss why the Artifacts are placed as they are.

The D-4 (Object Event Trace Diagram) is placed in both C2/R2 (column 2 / row 2) and C5/R3. The placing in C2/R2 is based on that D-4 identifies who is involved in the process that is performed, and the placing of D-4 in C5/R3 is the registration of the events that will happen during the process. The SA-9 (Web Application Diagram) is simply an abbreviation of an ESB in a SOA context.

The two EA Artifacts are not equal to the ESB Policy, but only interrelated. I see ESB Policy as a composite Artifact⁵⁴ in the *How* column covering the Contextual and Conceptual levels.

- The ESB-policy is a choice of having a central unit of integration, and thereby defines how the business should think of Services, and how the Services must interact - the Context level of the Zachman Framework.
- The definition of what responsibilities the ESB must support to ensure a common understanding of who has the responsibilities for which parts in completing a process. At this level there will be issues as:
 - Integration to legacy systems
 - Quality of Service (QoS) and Service Level Agreement (SLA)
 - Security and ESB management

These examples of ESB responsibilities might seem rather technical, and should be supported by the system. But with regard to the ESB Policy they are to be seen as principles of the ESB - a part of the definition on how the ESB is perceived by the organisation - the Conceptual level of the Zachman Framework.

As illustrated in the following figure there is only one place where the EA- and SOA Artifact share the same placing.

⁵⁴ An Artifact that spans over more than one cell in the Framework [EA using Zachman, p. 13]

	What	How	Where	Who	When	Why
Scope {Contextual}		ESB-policy				
Business Model {Conceptual}		ESB-policy, D-4				
System model {Logical}		SA-9			D-4	
Technology model {Physical}						
Detailed rep. {Out of context}						

Figure 34: ESB-policy EA - SOA mapping [Source: own work]

The ESB Policy is as noted not identical to the two EA Artifacts. The ESB Policy must be defined at a higher level which fits the placing in the *How* column. The placing of D-4 in the *When* column also fits with the previously discussed monitoring task of the ESB.

The purpose of mapping SOA Artifacts into the Zachman Framework is to illustrate the level of which I see SOA has a relevance to EA. The SOA Artifact “ESB Policy” is influencing the Contextual level of EA; the highest level of abstraction in EA when using the Zachman Framework. I am not arguing that SOA must be mapped into the Zachman Framework, and that this will solve your problems of the relations between EA and SOA. However, as discussed regarding SOA Artifact #3, there is need for a common framework of EA and SOA. However I do not see the Zachman Framework as an operational Framework that can encompass the complexity of both SOA and EA.

9.5.2 Version Control

Since the day it was possible to make an object that could be reused, versioning has been a challenge. In this context an “object” should be understood in its most general form - an example is making physical components for an assembly line.

Versioning is far from a new issue that has emerged with the introduction of SOA, and there is a considerable parallel to the world of Component Based Development (CBD), where the versioning issue is of great importance [114, p. 51]. I can not claim that SOA will make versioning more important than it was the case with CBD, simply because it in both cases is a necessity for success. SOA brings some changes for the reasoning and the view behind the versioning issue. In CBD versioning will often be a result of correcting errors, whereas in SOA versioning will be a result of changed requirements to the process. Secondly, in the CBD scenario a component will be something you implement into your system, whereas a Service in SOA is something you use. This is a simplification of the differences that exist, but illustrates the discipline of “versioning” changes with SOA. In Table 23 the SOA Artifact Version Control is therefore placed as both modifying and as a new Artifact.

My motivation for choosing the Version Control Artifact is however not only initiated by the fact that it will be where the change will be implemented (see section 9.5). but because I have seen so many examples of viewing versioning in a SOA context as being merely a technical issue [115][116][117][118] - in which I disagree. It is of course also a technical issue which must be solved, but by omitting to view the reason for the change requirements is in my opinion a great mistake. In order to not confuse the traditional understanding of the concept

of Version Control I will refer to this as Change Control as I see this more fitting in a SOA context.

In order to illustrate how complex change in a SOA context is I will play a little game called; The Beer Game. The Beer Game was initially developed to illustrate the need for distributing information in order to keep the optimal flow of goods in a supply chain [119, p. 28][120]. The parallel to SOA is that it is the change of demand that ripples through the supply chain. In SOA the scenario will be even more complex, as illustrated with the following figure.

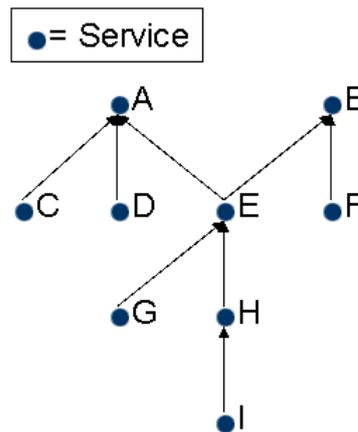


Figure 35 : The Beer Game of SOA [Source: own work]

Figure 35 is an illustration of how a set of nine Services are connected in order to provide the Services; A and B⁵⁵. What is interesting here is how a change in the demands on Service A will affect the rest of the Services. An example of a change of demand could be; the success of a Service.

A Service in a SOA context is an abbreviation of a process of the business, and one of the main goals of SOA is to reuse Services across the business which means that the Services will be used in new contexts for which they were not initially designed - reuse at its best! This will however entail performance issues as:

- Does the Service meet the non-functional requirements of the new usage scenario
- Will the increased request by new clients influence the performance supplied to existing clients - A possible SLA violation.

The above issues are only focusing on the issues at one level⁵⁶, but as illustrated in Figure 35 the requested Service can be dependent on a number of other Services which might not be able to support the new demands. The example is:

- Service A has become a success and now needs to support a greater number of requests without violating any SLA that *Service A* has with any other clients.
- This is not a problem for *Service A*, but *Service A* is dependent on *Service C*, *D* and *E*.
- The new requirements on *Service A* are within the capabilities of the SLA between *Service A* and *Service C*, *D* and *E*.
- No other *Service* will now need to know about the changes of *Service A* - *Service A* is running unchanged.
- However, now *Service B* becomes a great success, both inside and outside the Business.

⁵⁵ The Services C-I are also independent Services.

⁵⁶ One Service to a number of clients.

- The new demands on *Service E* are supported by the SLA between *Service E* and *F*, but are not supported by the SLA between *Service E* and *B*.
- It is necessary to change the SLA between *Service B* and *E*.
- This is however impossible until *Service E* changes it's SLA with *Service H*.
- This is also impossible until *Service H* changes SLA with *Service I*.
- After the SLA between *Service H* and *I*, *Service E* and *H* and *Service B* and *E* have all been updated, *Service B* can continue its success - if it is still needed.

The above example is about ensuring performance in accordance to the SLA's between the Services, but it could be many other issues than performance. What the example does not take into account are issues as; who will pay, how fast can this be done, security, data standards etc. The purpose here is not to show all the affects of change but to illustrate that if this is not done in an easy controlled manner SOA will never be agile!

Figure 35 is illustrated without an ESB as this essentially has nothing to do with the origin of the problem. However, the ESB can be part of the solution. With a set of nine Services it might be possible to manage the SLA's between the Services, but in a scenario of hundreds, or even thousands of Services there is an eminent need to automate as much as possible of this process. The ESB can monitor and foresee when Services need to be updated, and it can theoretically even update the SLA, but it cannot update the Services themselves.

This is the where the Change Control Artifact can define whom is responsible for which parts of the adoption to new requirements. So even though the Change Control Artifact is a sub Artifact of Quality Control it has direct references to the SOA Artifacts: 3.a Service Map, 8 Pricing Policy Model and 14 Enterprise Service Bus Policy.

Mapping to Zachman Framework

In Table 24 I did not map Version Control (Change Control) to any of Scott A. Bernard's EA Artifacts. However, the SOA Artifact Quality Control of which the Version Control is derived is mapped to SP-3: System Accreditation Document (See Appendix C). This EA Artifact is focused on security issues which is also one of the derived SOA Artifacts of Quality Control. Scott A. Bernard places the SP-3 Artifact in his EA³ Framework as depicted in the following:

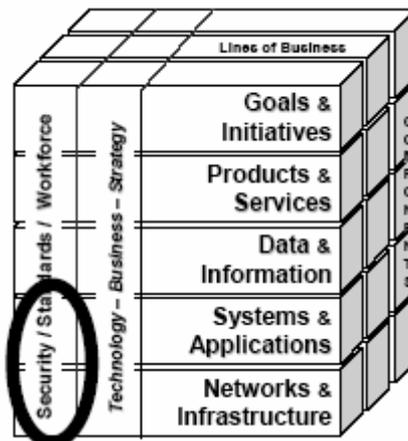


Figure 36 : Placing of SP-3: System Accreditation Document [Source: 110, p.40]

In Figure 36 the EA Artifact is placed in the cross level aspects of his EA³ Framework, which is perfect harmony in how I see the SOA Artifact of Change Control. I have in the previous described how the Change Control must be supported in many of the other SOA Artifacts, and if the agility of SOA is to be reached this goal must be aligned in a central strategy at all levels of the EA.

His mapping into the Zachman Framework is as shown in the following placed in C4/R5 :

	What	How	Where	Who	When	Why
Scope {Contextual}						
Business Model {Conceptual}						
System model {Logical}						
Technology model {Physical}						
Detailed rep. {Out of context}				SP-3		

Figure 37: Version Control (Change Control) - SOA mapping [Source: own work]

Scott A. Bernard does not argue for this placing, but from the description of the Artifact I find it difficult to believe that an issue that he places as a cross cutting element in his own model is “out of context” in the Zachman Framework. However, placing it in the *Who*-column fits not only with his SP-3 EA Artifact, but is also where I see the placing of the previously defined SOA Artifact: Change Control. The key findings from the previous section were:

1. Many aspects of SOA must be aligned with the Change Control.
2. A standard method must be developed to support the process.
3. As much as possible of the Change Control must be performed as automated processes, possible by the ESB.
4. As not all Change Control can be done automatically it is of the utmost importance that people are pointed out to control the change.

These points support Scott A. Bernard’s insertion in his own Framework, but will in the Zachman Framework, in my view, cover all the rows in column 4 and possibly even in other columns as well. Therefore I will not place the Change Control in the Zachman Framework.

As I have argued that Version Control is essentially the same as Change Control when in a SOA context. This fits with SOA being the paradigm that connects the business with IT - the changes in the business are changes in IT.

9.6 Summary

The purpose of this chapter was to make the relations of EA and SOA more concrete, and even start to speculate upon if EA and SOA will merge into to a new concept. My approach to analyse upon issues of which concrete parallels exists was based on EA Artifacts. In my perception this well known concept has a parallel in the context of SOA: The SOA Artifact. As the concept of a SOA Artifacts does not exist in the form I see it, I first of all had to define the concept. Secondly I used my experiences from working with SOA to identify and describe a set of these SOA Artifacts. I concentrated the effort on a high level of abstraction, but as well in identifying a set of derived SOA Artifacts. The interesting finding was that the higher the level of abstraction, the closer the resemblance to EA. This was perhaps expected from the findings of the previous chapters, but the important issue in this chapter was to make this clear on a more concrete basis.

I see the main value of Artifacts, both EA- and SOA Artifacts, lying in the relations between the Artifacts. Introducing SOA Artifacts amongst EA Artifacts will only make this issue even more complex, hence I focused on identifying the relations amongst the high level SOA



Artifacts. This clearly showed that essentially there are relations across all the Artifacts, and a methodology and/or Framework to support the “maintenance” of the Artifacts and relations are imperative to the success of SOA.

Finally I selected two of the SOA Artifacts and looked further at the characteristics of these and how they were related to specific EA Artifacts and the Zachman Framework. This “exercise” showed some of the complex problems of SOA, and that the Zachman Framework is not capable of grasping SOA in the long run.

The chapter began with the equation that: $SOA - A + EA = SOEA$. Looking at this in retrospect the question must be, if this was proven during this chapter? I cannot give a simple answer to this, and I will return to this in the next chapter. The reason I will not close this discussion in this chapter is that the findings of the other chapters must be included.



10 Reflection and perspective

I have throughout this thesis repeated the question if I am confusing the concepts of EA and SOA, and I am sure that many people do not share my perception of SOA. In order to get some assurance that I have not invented my own view on SOA I arranged an informal talk with Jakob Burkard who is Managing Consultant at Capgemini and a specialist on SOA to get the opinion of a practitioner. Besides of being an interesting talk we definitely shared the same views on SOA and EA - apparently quite a lot as I am now employed at Capgemini.

Throughout this thesis the main focus has been to identify parallels and relations between SOA and EA. I have done so approaching the matter from different angles in order to get the broadest possible picture. All of which essentially has been done to show how I see the evolution of SOA and EA.

In this last chapter I will sum up on the key findings of what you have read so far, but more importantly I will look at how these findings support each other in order to answer the thesis problem.

10.1 Chapter Summary

I will first shortly recap the contents of what I have discussed so far:

Chapter 5: The Hype of SOA

In short this chapter was about showing how the concept of SOA is “sold”. The reason that I find the subject of this chapter to be relevant is that I see the hype of SOA as being the reason behind many of the misconceptions behind SOA - discussed in the later chapters.

Chapter 6: Background and concepts

This can be viewed as the theoretical basis of the thesis. This however differs somewhat from what is the normal approach of writing the theoretical foundation in a thesis. The chapter is based on the assumption that the reader is already familiar with the concepts, hence the main focus is to clarify how I interpret the concepts of SOA.

The reason for this approach is divided in two:

1. It would be out of scope for this thesis to fully explain the concepts - each of the concepts could probably be the topic of a thesis.
2. In the world of SOA many different interpretations of the concepts exist, hence to avoid misunderstandings it is always necessary to define the basic concepts - this was also seen as a SOA Artifact in chapter 9.

Chapter 7: Evolution and maturity of SOA and EA

This chapter had two subjects on the agenda. First I needed to elaborate on why I see SOA as I do, and fit this into a bigger picture. This was done by discussing the evolution as a general issue in the world of IT and using this to look of the concrete evolution of SOA and EA.

Another way of looking at evolution is through the concept of maturity, and as this is a well established concept in the world of EA, I used a model developed by Peter Herzum in order to identify parallels between SOA and EA. This lead to a very interesting discovery: EA at its most mature state equals the definition of SOA.

Chapter 8: Perspectives of SOA's impact on EA

Where chapter 7 looked at the relations between SOA and EA mainly from a SOA perspective, I changed the perspective in this chapter to that of EA. I based this work on an article by Boris Lublinsky and Dmitry Tyomkin; “Dissecting Service-Oriented Architectures”. As such the article only superficially touched the subject, and what they missed was the important relations that must be handled between the Artifacts



Chapter 9: Service Oriented Enterprise Architecture

Until this chapter the relations between SOA and EA had primarily on a conceptual level, but as I pointed out in my motivation of this thesis it is in fact a big problem for the people who should harvest all the promised benefits of SOA and EA - the practitioners! I will by no means claim to have solved this issue, but as I see my self as a practitioner I used myself as a frame of reference; asking where I would like to see concrete examples of where SOA and EA interconnect.

It is my claim that SOA, like EA, will create a bundle of Artifacts, and I used this issue as the focus point of identifying concrete parallels and relations between SOA and EA. I identified a set of SOA Artifacts and compared these with a set of EA Artifacts identified by Scott A. Bernard.

The result of this was that the high level SOA Artifacts would only modify existing EA Artifacts, whereas the derived SOA Artifacts would more often add completely new aspects into the frame of EA. I started the chapter by advocating that SOA and EA would merge into a new concept, but in fact left the follow up on this issue to the following chapter - which you are currently reading.

10.2 The evolution

As described in section 6.1.1 EA is founded from a business perspective and as discussed throughout this thesis SOA is founded from a technology perspective. A consequence of these different perspectives has been that EA is typically seen as a Top-Down approach, and SOA is seen as a Bottom-Up approach - resulting in very different types of projects. This was discussed in section 7.1, where the following table was used to illustrate the differences of the two approaches.

Bottom-Up Approach Point Projects	Top-Down Approach Area Projects
Local short-term initiative	Broader, longer-term initiative
Building a solution against immediate requirements (where “building” means design, construct or assemble)	Focus on system properties across a whole area (e.g. business domain, technical domain, infrastructure)
Strongly aligned to local objectives.	Creating value by establishing (procuring or building) conveniently available resources
Cost-effective use of conveniently available resources (improvisation or “bricolage”)	Indirect links between benefits (across area), costs and risks
Direct link between (local) benefits, costs and risks.	<ul style="list-style-type: none"> Often difficult to create/maintain business case for adequate investment in resources and infrastructure
No mandate to pay attention to broader, longer-term opportunities and effects.	<ul style="list-style-type: none"> Often difficult to demonstrate return on investment

Table 25 : Bottom-Up versus Top-Down [Source: 109]

In section 7.1 I did not discuss which of the two is preferable - which would also be a question with no single answer. Both the Top-Down and Bottom-Up approach has qualities and weaknesses. The Top-Down approach has its advantages in that it will create cohesiveness across the enterprise, whereas the Bottom-Up approach has its strength in the detail of the specific context. So, what you essentially want is the best of both.

This thesis has strongly advocated for seeing SOA as being Top-Down, such as EA. However this view is also discussed in relation to the evolution of SOA, which is a central issue of this thesis. The reason for this being seen as so important is that the evolution of SOA has mainly been focused on the heightening of the level of abstraction. The consequence of this has been that SOA has entered issues which previously were covered in the domain of EA.

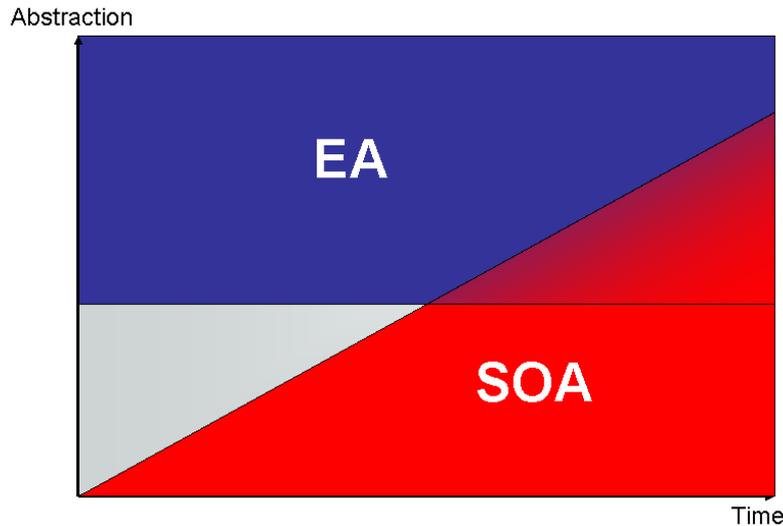


Figure 38 : Crossover of EA and SOA [Source: own work]

Figure 38 illustrates that SOA and EA have evolved in a way that there is a common area covered by both of them. This crossover is bound to entail conflicts on where issues of this area belong. It is this area of conflict which is the base of why this thesis has its relevance. If one looks aside the conflicts what the is also illustrated in Figure 38 is that all the levels of abstraction are covered by the combination of EA and SOA, meaning that if the conflicts are solved a holistic view of the business at all levels of abstraction can be achieved.

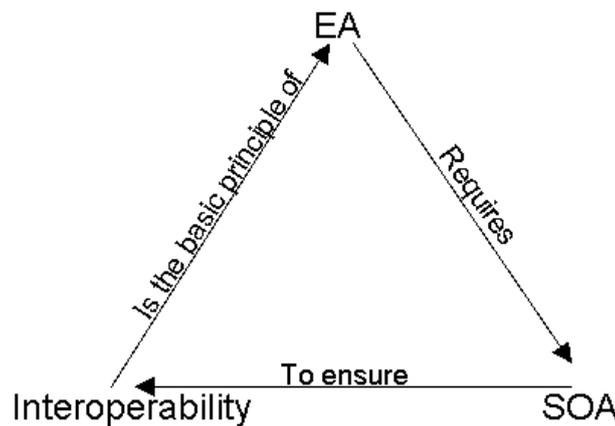


Figure 39: The link between EA, SOA and Interoperability [Source: 126 – translated from danish]

Figure 39 is a good illustration of how the concepts of EA, SOA and Interoperability are related. The strength of the illustration is that it is not dependent of which of the three aspects is your starting point. There is however an issue of the illustration that does not fit my view on how SOA and EA affect each other; as previously stated I also see that SOA requires EA. I confronted Jakob Burkard, who is one of the authors, with my view. He agreed that SOA indeed also needs EA, hence resulting in the following updated version:

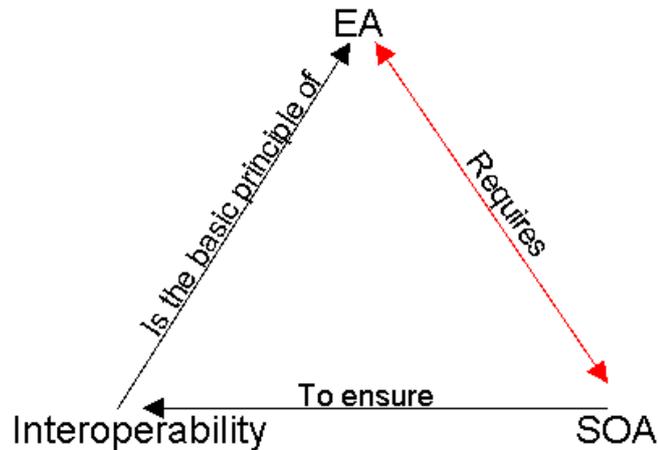


Figure 40 : The link between EA, SOA and Interoperability – updated version [Source: own work]

As discussed in section 9, the practical SOA projects seem to have a tendency to evolve into having more aspects that normally are placed within an EA program. As promised I will return to this using the example of Nykredit where I, based on my interview with Mikkel Haugsted Brahm, see their project as going through the following stages, as illustrated on Figure 41 - note that the arrows points in the opposite direction than illustrated on Figure 41:

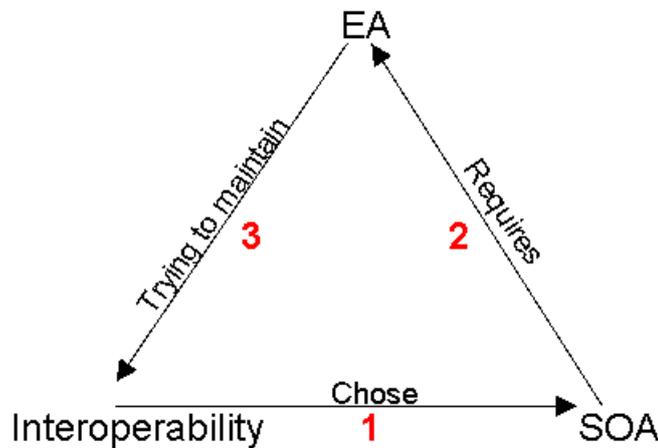


Figure 41 : The steps of Nykredit's SOA project [Source: own work]

First of Nykredit chose Component Based Development to better organise- and improve reuse of their software portfolio. By coincidence the concept of SOA arose and promised to solve the interoperability issues. They felt that their original CDB strategy fitted well with this paradigm, and took their first step down the SOA road (as illustrated on Figure 41). They have now gained knowledge of SOA and have experienced issues that does not fit within their perception of SOA. This is their current situation and it is my guess that they will seek possible answers in the “world” of EA. This is illustrated as step 2 on Figure 41, and they are probably able to find aspects from EA that can help handling some of the issues that have risen from SOA. I see this approach as being very risky, as the possible consequence is that you will be trying to maintain interoperability issues which was originally the purpose of introducing SOA, hence you will be prone to conflicts between SOA and EA.

This example is of course not necessarily what will be the case. But what it illustrates is the importance of seeing EA and SOA in a cohesive whole!



10.3 Service Oriented Enterprise Architecture

As stated by Gurpreet S. Pall that; "If you don't do EA, you can't do SOA". The value of SOA will not be realizable before it is closely integrated in EA. On the other hand that SOA can be the long needed methodology that can make EA operational! Therefore seeing SOA and EA as two separate issues will mean that the possible synergy will never be obtained.

I introduced the concept of SOEA in chapter 9 using the equation: $SOA - A + EA = SOEA$. As illustrated in Figure 42 the EA process starts before the SOA process, which is because I essentially see the decision of "going SOA" as being a decision of EA.



Figure 42: Service Oriented Enterprise Architecture [source: own work]

Figure 42 is a graphical representation of how I see the future of EA and SOA. EA will still provide the big picture and SOA can be viewed as a Plug-In to EA. The result will in my vision provide a stronger integration between the business and IT through what I call: Service Oriented Enterprise Architecture (SOEA). In chapter 9 the concept of SOEA was left to be elaborated, which is the focus of the rest of this section

In chapter 7 one of the key findings of this thesis; that the definition of EA at its most mature state (Nirvana) is equal to my definition of SOA. An equality which is aligned with the shared goal of SOA and EA; only needing to optimize the business and having IT seamlessly supporting this - indeed a state of Nirvana. In section 7.5 some very interesting issues on where the different stages of maturity had their focus was discussed, which I have illustrated in the following.

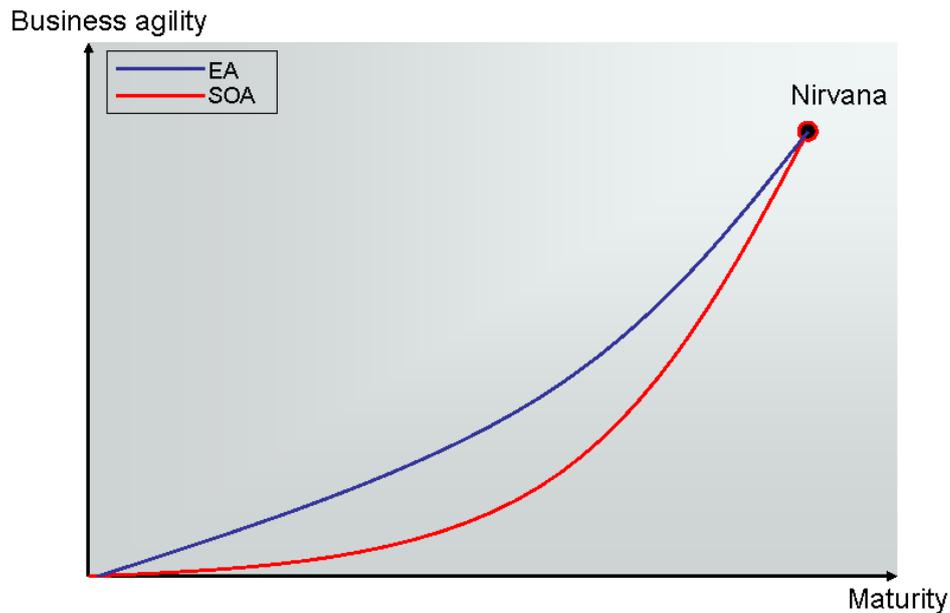


Figure 43 : The "road" to Nirvana [Source: own work]

Figure 43 shows that SOA and EA are not using straight paths to reach the state of Nirvana. As I discussed in section 7.5 the EA maturity model developed by Peter Herzum can be applied on both EA and SOA - This is illustrated on the x-axis on Figure 43. The y-axis on Figure 43 is representing a more abstract concept of Business agility. It must be noted that Figure 43 is for illustration purpose only and that the inclination of the curves are not based on empirical values. Their purpose is to illustrate that neither EA nor SOA will be the shortest path to Nirvana. The reason that SOA is placed below EA is because it focuses on the technical agility whereas EA has a stronger focus on the agility of the business.

EA and SOA has in my opinion little value by them selves, when compared to the value you get from combining EA and SOA into one: SOEA.

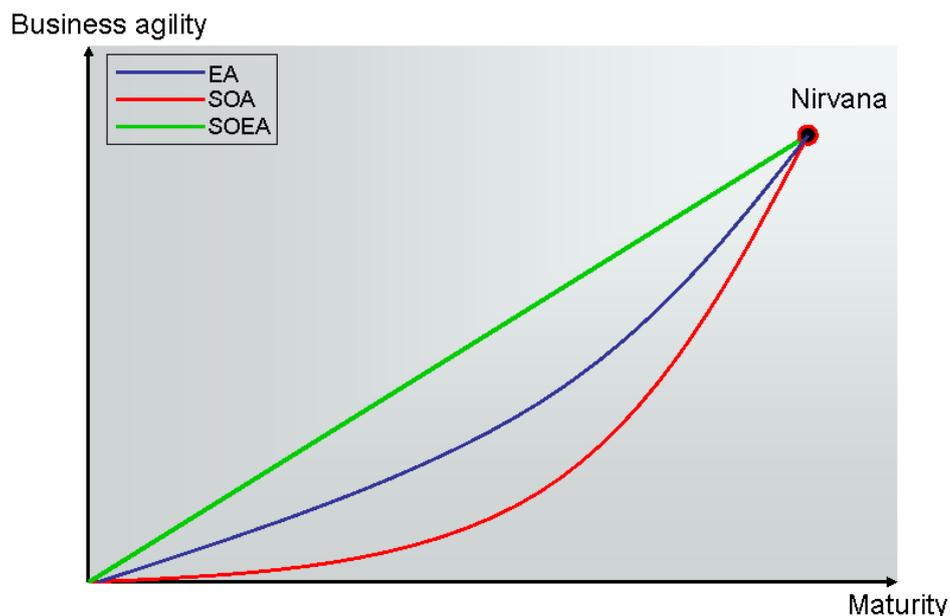


Figure 44 : The "SOEA-road" to Nirvana [Source: own work]

Figure 44 adds SOEA to Figure 43; as the "straight road" to Nirvana. What is important to remember is that no enterprises have yet reached the maturity state of Nirvana - making the

illustration somewhat theoretical. The identification of the shared goals of EA and SOA is however very important as this gives the possibility of using the best capabilities of EA and SOA in order to reach the common goal. There is no practical evidence of this, but this has been the sole purpose of this thesis to identify the parallels between EA and SOA in order to prove that EA and SOA must be seen together. I have identified the relations on a conceptual level and proceeded to identify the relations on a more practical level. So why is it the combination of EA and SOA that can provide this symbiotic effect? The answer in fact lies in one of the motivations behind both SOA and EA; aligning resources across the enterprise.

10.4 SOEA process

In section 6.2.2 the SOA Process was discussed. The SOA Process was also seen in relation to EA in general using the work of Scott A. Bernard. This was the first occurrence where the relation between SOA and EA was discussed. The following section of the chapter did not focus further on the subject but was to set a common theoretical ground on which a collective view on SOA and EA could be based. I have introduced the concept of SOEA and will in this section look at the process of SOEA.

Maturity models of EA are often seen as going through a process. This parallel is also interesting to map onto SOA. However as described section 7.5 Herzum's maturity model is also applicable for SOA, but EA and SOA will differ in the level of abstraction throughout the maturity process

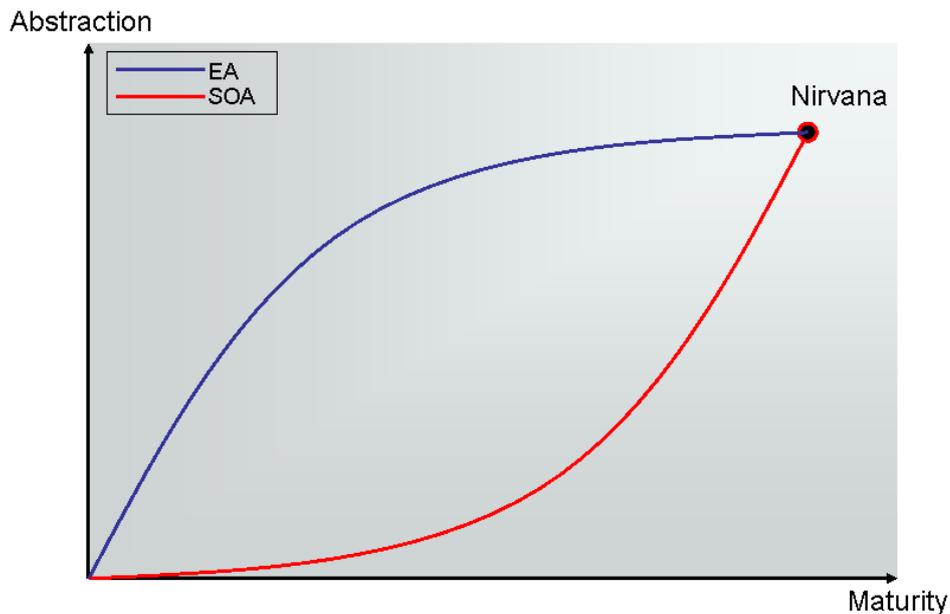


Figure 45 : Abstraction and maturity of EA and SOA [Source: own work]

Figure 45 is a graphical illustration of what was discussed in section 7.5, relating the maturity level to the level of abstraction. The illustration shows, as Figure 43, the very different paths to the state of Nirvana.

These different paths are based on the notion that EA and SOA are seen as two different projects within the enterprise originating in different worlds (see section 6.1.1), and here lies thy key! It is eminent that these paths must be aligned onto a common path. The focus of chapter 9 was to identify concrete parallels of EA and SOA using the concept of SOA Artifacts. Identifying these SOA Artifacts does not only demonstrate specific parallels but they can also be used to align the process of EA and SOA.

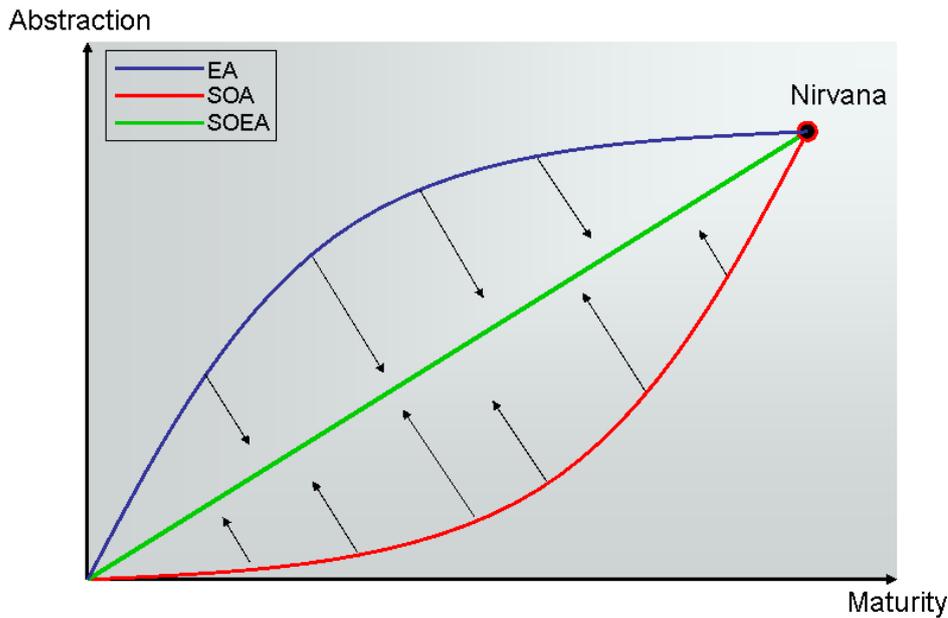


Figure 46 : Abstraction and maturity of SOEA [Source: own work]

Figure 46 illustrates how the alignment of EA- and SOA Artifacts can help align the maturity process of EA and SOA - establishing a common base of SOA and EA. As discussed in section 10.3 the conflict of SOA and EA is due to the fact that both operate in the same level of abstraction, but by aligning SOA and EA and developing a common process will change the picture:

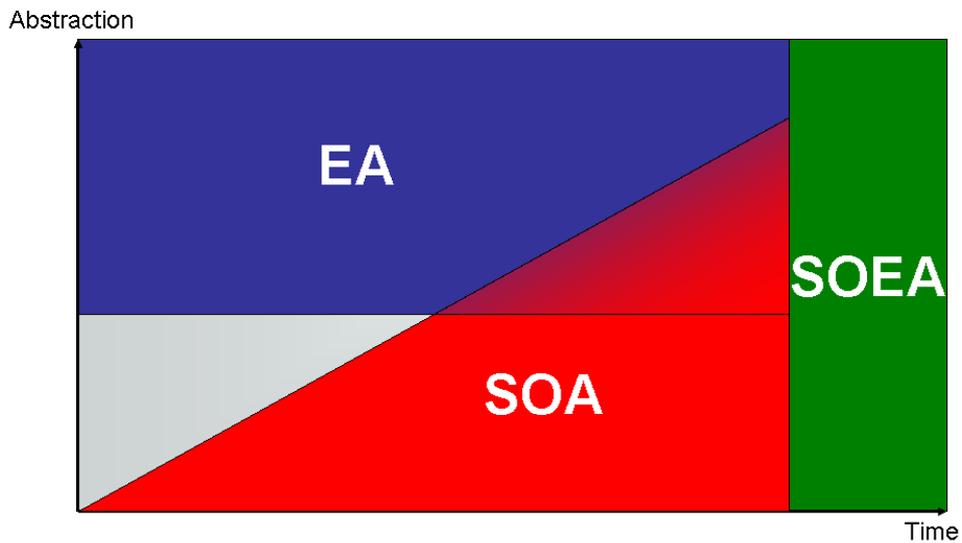


Figure 47 : SOEA - the alignment of SOA and EA [Source: own work]

I will not discuss the development of a SOEA Process any further. The purpose here is to show that a common understanding of the conflicting areas is the key to correlate SOA and EA and obtain a synergy.

10.5 SOEA Governance

EA without governance and vice versa will make little sense - a statement which can be repeated concerning SOA and governance. In section 9.2 I identified the SOA Artifact titled "SOA Governance", but have otherwise omitted to discuss this issue during this thesis. This has been a deliberate choice as the issue of governance would be out of scope of this thesis.



That being said, I will briefly touch the subject. The governance issue must be a big part of the SOEA Process, and in parallel to this, the need of a common frame of reference between SOA and EA is a prerequisite for SOEA Governance to be successful. Is SOEA Governance therefore the same as I identified in the SOA Artifact “SOA Governance”? The answer here is in fact a very good illustration of why I have written this thesis. The purpose of this thesis was not to define a method to integrate the business with IT and create the “Agile enterprise”. The purpose has been to set a common ground for this work, and Governance is definitely a part of this work, and get a better understanding of the relations between SOA and EA.

SOEA Governance is the symbiotic effect correlating SOA Governance and EA Governance. What SOA brings to the traditional discipline of EA Governance is that much of the governance can be automated. One of the means here is the ESB (as discussed in section 9.2), but in order to utilize the ESB the enterprise must carefully define their ESB Policy. Besides the parts that can be automated SOA will also bring many tasks to the SOEA Governance. This was also identified in section 9.2 where several of the SOA Artifacts required staffing.

The issue of Governance and the relation between SOA and EA is in fact also an issue that is raised by Forrester. In the article “What Must EA Do To Sustain SOA?” the following quote

“This is where second-generation SOA begins – where SOA is a given and the focus is on how IT supports the business. EA will continue to drive the way that enterprises utilize SOA, but this leadership will be different from how EA has championed SOA thus far.”

[128]

I see the change in how EA champions SOA being of such a character that we need to consider if we need to re-evaluate almost all issues of SOA and EA. If we call it Service Oriented Enterprise Architecture is of minor importance, but I see the work of this thesis as a good common base for the future work.

10.6 Summary

Looking back at all the issues that have been raised throughout this thesis it might seem questionable if the promises listed in chapter 5 are achievable. Sometimes I have even wondered if SOA is just a reinvention of all the classical problems of computer science, just to keep the consultants busy - it certainly will for a long time to come.

However there is a good reason to “reinvent” the classical issues of computer science, as the focus has changed. We are building on a good knowledgebase of OO and CBD, and is now trying to make standards which can solve the problems regardless of the given platform. This evolution of SOA has lifted the concept from merely being a technical concept to be the paradigm of how to do business - lifting the level of abstraction to a level which until now has been solely covered by EA.

Throughout this thesis I have identified many of these correlating issues. It is my statement that, if these issues are left to be handled separately by EA and SOA, this will be a source of conflict that will jeopardize the entire enterprise - a common base must be established to solve this issue.

The question is what this common base is. In the problem of the thesis the question was asked, if SOA and EA have a spurious correlation, or is there something causal between the two?

The causal relation was refuted as SOA will not cause EA, and vice versa. One could argue that the example of Nykredit now looking at EA was a causal result of the choice of SOA (SOA → EA), but I will omit this as this was unintentional, and furthermore I have identified that relation is bidirectional.

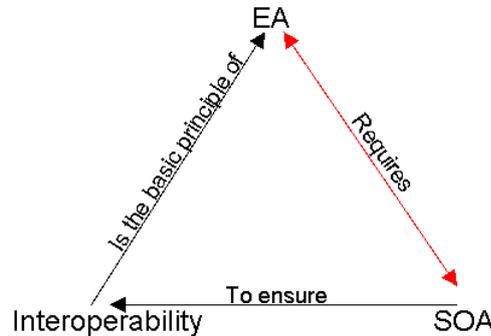


Figure 48 : The link between EA, SOA and Interoperability – updated version [Source: own work]

This was illustrated in Figure 40 (repeated above), and the creator of the original model (Figure 39) agreed that the dependency of EA and SOA is bidirectional.

The answer lies in the spurious correlation of SOA and EA. There are several of these, but the most important is that SOA has evolved and is now, at some levels of abstraction, operating on the same level of abstraction as EA. One solution to this problem would be to establish a clear line between SOA and EA. However in doing so, it is my prediction that the gap between the business and IT will remain as experienced with EA. This must be avoided at all costs, which is where I see SOEA.

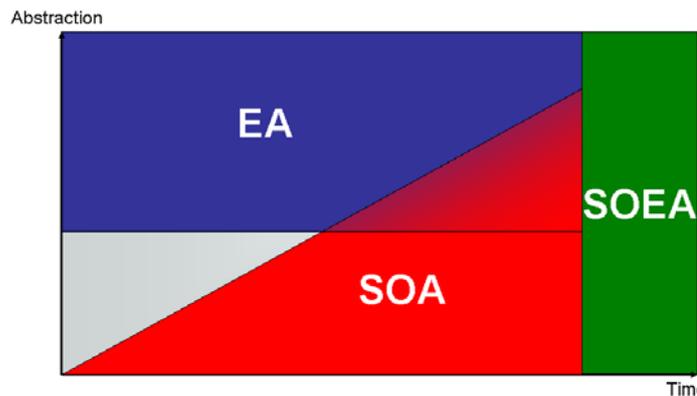


Figure 49 : SOEA - the alignment of SOA and EA [Source: own work]

Figure 49 (as depicted in Figure 47) SOEA is built on the experiences of SOA and EA, and this thesis is the first step in defining SOEA. The difference from establishing a clear line between SOA and EA, is that SOEA must cover all the levels of abstraction previously covered by SOA and EA, but the SOEA must do so in a collective hole.

The purpose of this thesis has not been solving the problems of either SOA or EA. The purpose has been to set the base for the continued evolution of SOA and EA, and much work is needed. But I will claim that the work can continue on a solid base, and I am hoping to be a part of the next chapter of the evolution of SOEA.



11 Conclusion

SOA has in the last couple of year evolved at a tremendous rate. SOA was originally founded as a technical concept but is now a concept that is closely tied to the enterprise. As a result, SOA is now operating at an abstraction-level far from the technical origin.

As SOA starts to operate at this high level of abstraction it enters an area which until now has been covered by EA. This entails a conflict of responsibility between SOA and EA as the coordination between the two is inadequate.

I have identified that SOA and EA have shared goals, which could indicate a causal relation between the two. I have however refuted this as the strength of SOA and EA originates from the combination of the two - they must not be seen as separate processes.

Today SOA and EA are commonly approached as two separate disciplines, but the analysis of this thesis shows that the relations between SOA and EA are of such an extent that this not a feasible approach. That SOA and EA share the same goals is an indication of this, but that the definition of EA at its most mature state is equal to the definition of SOA is one of the key findings of this thesis. In the world of EA, Maturity Models are a well known concept, but is a concept only now emerging in the world of SOA. Having separate Maturity Models of SOA and EA is however insufficient, if the desire is to align SOA and EA. The paths of SOA and EA to Nirvana must not be separate paths; they should be aligned in order to get a symbiotic effect of SOA and EA.

SOA is often referred to as being what can make EA operational. This I believe is true, but the changes that EA will endure as a consequence of introducing SOA must entail that the evolution of SOA and EA is seen as a combination of the two. I have argued that aligning SOA and EA will have such an influence on both SOA and EA that we need to view the two as a new concept: SOEA.

The focus of this thesis have been to set a common base on which this alignment can be grounded. The common goal might seem as the obvious choice, but I believe that the answer lies in the spurious correlation of SOA and EA: That SOA and EA are operating on the same level of abstraction is the source of many conflicts and much confusion. However if approaching this as a spurious correlation puts attention on the subject, and it is my belief that the common level of abstraction is in fact the key to couple the business with IT.

My motivation of this thesis was that I needed to make the connection between SOA and EA more explicit, and identifying spurious correlation as the common level of abstraction is only an explanation on why confusion exists. The solution was to use the concept of EA Artifacts, and use the same concept on SOA (SOA Artifacts). This very clearly demonstrated the consequences that SOA has on EA; SOA does not only add new SOA Artifacts, it will create SOA Artifacts that cover an area which would normally be covered by EA Artifacts. This clearly demonstrates that if SOA and EA are not thought of as one, the relations between the Artifacts of SOA and EA will be impossible to maintain over time. It is my belief that the alignment SOA- and EA Artifacts (SOEA Artifacts) into a common frame of reference is evident for the continued success of both SOA and EA.

SOEA should not be viewed as a complete package to solve the problems of integrating SOA and EA to a cohesive whole. Much work is needed, where the issues of Governance and a Documentation Framework are obvious areas needing more work. These issues have been shortly discussed within the content of this thesis, but the work needed here is far beyond the scope of this thesis. This thesis sets the base of this work, and I hope that I can continue the work to closer integrate the business with IT - creating the agile enterprise.



12 Aftermath

This section has been removed due to confidential content.



13 Appendix

13.1 Appendix A

The following is a collection of definitions on the Concept of SOA [34]:

"SOA is a framework enabling application functionality to be provided, discovered and consumed as re-usable Web Services sets. While Web Services do not equal SOA, it's one of the enabling standards. SOA abstracts complexity and implementation details, making it an ideal architectural mindset to leverage functionality trapped within mainframe/midrange systems."

Scott Rosenbloom is chief strategist with WRQ Inc.

"Secure, integrated delivery of IT solutions meeting business requirements. Solutions must implement, optimize and guide business process execution by combining the functionality of separate, discreet, reusable services. SOA moves away from complex application development, promoting a focus on standardizing interfaces between atomic service components with centralized management and distributed implementation."

Dave Morris, I.T. Security Lead TransAlta Corp.

"The SOA models the business as a collection of self-contained services that are available across the enterprise that can be evoked through standard protocols both internally and externally."

Dave McComb, president, Semantic Arts

"Service Oriented Architecture is nothing but business oriented architecture, which allows the flexibility of business applications, to become independent but collaborative, while providing their services. The applications under this architecture are both 'client' and 'server' at the same time with freely available services."

Satheesan Kunnel, USWWI

"A service oriented architecture is an approach to design and integrate software in a modular method where each module is precisely a 'loosely coupled service' that is accessible over a network and has the capability of being dynamically integrated with other services at run time. A service must present a standard Interface (be it WSDL today) for its functionality and invocation methods while the real implementation of the service is not a concern of an SOA."

Rajesh Dawar

"Services provide something of value to those who know how to request and consume them, without having to know how to produce that value. SOA is an approach to building software applications as collections of autonomous services that interact without regard to each other's platform, data structures, or internal algorithms."

Michael Champion, R&D specialist, Software AG



"A pattern of design, development, deployment, and management of (a) applications and (b) software infrastructure and frameworks in which:

- Applications are organized into business units of work (services) that are (typically) network accessible
- Service interface definitions are first-class development artifacts
- Quality of service (QoS) characteristics (security, transactions, performance, etc.) are explicitly identified at design time
- Software infrastructure takes active responsibility for managing QoS and enforcing policy for service access and execution
- Services and their metadata are cataloged in a repository
- Protocols and structures within the architecture are, optionally, based on industry standards (e.g., the emerging SOAP stack of standards)

Randy Heffner, vice president, Forrester Research Inc.

"SOA is a style of design that strives to enable easy integration and flexible applications. In SOA, application functionality is designed as shared reusable services. A service is a piece of application functionality that exposes its functionality through an abstract interface, which hides the inner workings of the service implementation."

Anne Thomas Manes, analyst, Burton Group

"A SOA is an enterprise-scale architecture (typically spanning multiple applications within an enterprise or across multiple enterprises) where the primary structuring element is a service (as opposed to modules, systems, applications or components).

A service is a set of related business functions that are interacted with locally or remotely using a message-passing/document-oriented communication style. A service is composed of (1) a (functional) service interface and (2) a service implementation that implements one or more service interfaces and adheres to a certain set of (non-functional) capabilities. Specific services are defined in terms of the transport/application/messaging protocol, not in terms of a specific programming model.

A SOA will typically include technical services to manage metadata about service interfaces and implementations, service providers and service consumers; and services for managing and enforcing policies, access control, security features, and transactions, although all of these are optional within any specific SOA instance."

Stefan Tilkov, CEO, innoQ

"Service-oriented architecture is an architectural discipline that centers on the notion that IT assets are described and exposed as Services. These Services can then be composed in a loosely-coupled fashion into higher-level business processes, which providing business agility in the face of IT heterogeneity."

Ronald Schmelzer, analyst, ZapThink LLC

"Service Oriented Architecture (SOA) is an approach to the development of loosely coupled, protocol-independent distributed applications composed from well-defined, self-contained software resources accessible as Services across the extended enterprise in a standardized way, enhancing re-usability and interoperability."

Ankur Gupta, marketing manager, Fiorano Software Inc.



"SOA is a form of technology architecture that adheres to the principles of service orientation. When realized through the Web Services technology platform, SOA establishes the potential to support and promote these principles throughout the business process and automation domains of an enterprise."

Thomas Erl, chief architect, XMLTC Consulting Inc



13.2 Appendix B

List of members of of WS-* stack [82]:

[WS-Addressing]

Web Services Addressing (WS-Addressing). Don Box, et al. August 2004. BEA, IBM, and Microsoft.

[WS-AT]

Web Services Atomic Transaction (WS-AtomicTransaction). Luis Felipe Cabrera, et al. September 2003. BEA, IBM, and Microsoft.

[WS-BA]

Web Services Business Activity Framework (WS-BusinessActivity). Luis Felipe Cabrera, et al. January 2004. BEA, IBM, and Microsoft.

[WS-Coord]

Web Services Coordination (WS-Coordination). Luis Felipe Cabrera, et al. September 2003. BEA, IBM and Microsoft.

[WS-Discovery]

Web Services Dynamic Discovery (WS-Discovery). John Beatty, et al. February 2004. Microsoft Corporation.

[WS-Enum]

Web Service Enumeration (WS-Enumeration). Don Box, et al. September 2004. Microsoft Corporation.

[WS-Eventing]

Web Services Eventing (WS-Eventing). Luis Felipe Cabrera, et al. September 2004. BEA, Microsoft, and TIBCO.

[WS-Federation]

Web Services Federation Language (WS-Federation). Siddharth Bajaj, et al. 8 July 2003. IBM, Microsoft, BEA, RSA Security, and VeriSign.

[WS-FedActive]

WS-Federation: Active Requestor Profile. Siddharth Bajaj, et al. 8 July 2003. IBM, Microsoft, BEA, RSA Security, and VeriSign.

[WS-FedPassive]

WS-Federation: Passive Requestor Profile. Siddharth Bajaj, et al. 8 July 2003. IBM, Microsoft, BEA, RSA Security, and VeriSign.

[WS-MEX]

Web Services Metadata Exchange (WS-MetadataExchange). Keith Ballinger, et al. March 2004. BEA, IBM, Microsoft, and SAP.

[WS-Policy]

Web Services Policy Framework (WS-Policy). Don Box, et al. 3 September 2004. BEA, IBM, Microsoft, and SAP.

[WS-PA]

Web Services Policy Attachment (WS-PolicyAttachment). Don Box, et al. 3 September 2004. BEA, IBM, Microsoft, and SAP.



[WS-RM]

Web Services Reliable Messaging (WS-ReliableMessaging). Ruslan Bilorusets, et al. March 2004. BEA, IBM, Microsoft, and TIBCO.

[WS-SecureConv]

Web Services Secure Conversation Language (WS-SecureConversation). Steve Anderson, et al. May 2004. BEA Systems, Inc., Computer Associates International, Inc., International Business Machines Corporation, Layer 7 Technologies, Microsoft Corporation, Netegrity, Inc., Oblix Inc., OpenNetwork Technologies Inc., Ping Identity Corporation, Reactivity Inc., RSA Security Inc., VeriSign Inc., and Westbridge Technology.

[WS-Security]

Web Services Security: SOAP Message Security (WS-Security). Ed. Anthony Nadalin, et al. March 2004. OASIS-Open.org.

[WS-SecurityPolicy]

Web Services Security Policy Language (WS-SecurityPolicy). Giovanni Della-Libera, et al. 18 December 2002. IBM, Microsoft, and VeriSign.

[WS-SecUsername]

Web Services Security: Username Token Profile V1.0. Ed. Anthony Nadalin, et al. March 2004. OASIS-Open.org.

[WS-SecX509]

Web Services Security: X.509 Token Profile V1.0. Ed. Phillip Hallam-Baker, et al. March 2004. OASIS-Open.org.

[WS-Transfer]

Web Service Transfer (WS-Transfer). Ed. Don Box, et al. September 2004. Microsoft Corporation.

[WS-Trust]

Web Services Trust Language (WS-Trust). Steve Anderson, et al. May 2004. BEA Systems, Inc., Computer Associates International, Inc., International Business Machines Corporation, Layer 7 Technologies, Microsoft Corporation, Netegrity, Inc., Oblix Inc., OpenNetwork Technologies Inc., Ping Identity Corporation, Reactivity Inc., RSA Security Inc., VeriSign Inc., and Westbridge Technology, Inc.

13.3 Appendix C

EA ³ Framework Area	Artifact # and Name		
	<p>D-4: Object Event Trace Diagram</p> <p>Also called an Object 'Sequence' Diagram, the D-5 diagram allows the tracing of actions in a set of scenarios or operational threads. Each model should focus on a critical sequence of events and a description of this scenario should accompany the model.</p>		
Example			
<p>With time proceeding from the top of the diagram to the bottom, a specific diagram lays out the sequence of information exchanges that occur between business nodes for a given scenario. These information exchanges are associated with events and actions (see Information Exchange Matrix). The direction of the event arrows shows flow of control, in terms of the business process, from node to node.¹</p>			
¹ K.Sowell and A. Reedy, 2001			
Relationship to Other EA Frameworks			
FEAF: Data Level	FEA: DRM	Zachman: C2/R2, C5/R3	DODAF: OV-6c, SV-10c

EA ³ Framework Area	Artifact # and Name		
	<p>SA-9: Web Application Diagram</p> <p>The web application diagram shows the logical relationships between web-based information services, in this case showing a detailed diagram of services that interact via standard protocols and interfaces that promote platform-independent data interchanges.</p>		
Example			
Relationship to Other EA Frameworks			
FEAF: Application Level	FEA: SRM, TRM	Zachman: C2/R3	DODAF: SV-2

EA ³ Framework Area	Artifact # and Name		
	<p align="center">SP-3: System Accreditation Document</p> <p>The System Accreditation Document uses a standard format for evaluating the security status of information systems throughout the enterprise. There are a number of parts to a system security accreditation as are illustrated in the example.</p>		
Example Outline			
<ol style="list-style-type: none"> <u>System Security Plan</u>. This opening section of the System Accreditation Document provides an overview of the business context that the information system operates in, states the current security status of the system (last accreditation), and summarizes the contents and finding of the other accreditation documents. <u>System Risk Assessment</u>. This section of the document uses a standardized format for showing areas of risk to the information system in the four primary areas security threat areas that are covered in artifact SP-2; physical, data, operational, and personnel. Assigns a level of risk based on the business context for system operations and the type of system data to be protected. Provides security risk remediation strategies (how to avoid a security risk, or deal with it if a problem occurs) for each area of risk that is identified. <u>System Test and Evaluation</u>. Also called a system 'penetration test.' The System Test and Evaluation (ST&E) section of the document provides the results of a live test that attempts to enter the system through other-than-normal log-in procedures, as well as attempts to overwhelm the system (denial of service attack), or infect the system with an active virus, worm, or other type of problematic element that reduces or eliminates information system functionality. <u>Remediation Plan</u>. This section of the document provides the status of corrective actions taken to fix all of the security risks found during the risk assessment/ST&E. <u>Approval to Operate</u>. This section of the document is the formal (signed) approval to operate the information system that is provided by the designated person in the enterprise (usually the Chief Information Officer or the IT Security Manager). 			
Relationship to Other EA Frameworks			
FEAF: None	FEA: SPP	Zachman: C4/R5	DODAF: None



13.4 Appendix D

Enterprise Resource Planning is defined as: An enterprise-wide set of management tools that balances demand and supply, containing the ability to link customers and suppliers into a complete supply chain, employing proven business processes for decision-making, and providing high degrees of cross-functional integration among sales, marketing, manufacturing, operations, logistics, purchasing, finance, new product development, and human resources, thereby enabling people to run their business with high levels of customer service and productivity, and simultaneously lower costs and inventories; and providing the foundation for effective e-commerce [89, p. 5]



13.5 Appendix E

Summary of interview with Mikkel Haugsted Brahm Nykredit.

This section has been removed due to confidential content





13.6 Appendix F

As the methodology of the theses was exploratory the level knowledge on the subject of writing will of cause evolve throughout the process. The obtained knowledge has led me to change my thesis problem from the originally posted:

Managing the evolution of complex information systems (IS) is by no means a new ordeal. Service Oriented Architecture (SOA) has often been proclaimed to be the “silver bullet” to eliminate this problem. But evolution can come in many shapes and levels of an enterprise. SOA is the natural evolution in software development, enabling a higher level of abstraction, which has made it easier to lift IT to a management level. This higher level of abstraction, I believe, has also resulted in the need for a new coupling between the more physical development and the architectural design of software systems. This becomes clearer when an implementation based on SOA has existed for some time, and the coupling between the different layers of abstraction must be maintained.

The purpose of this thesis is to identify and describe:

- The problems that arises when an implementation based on SOA grows and change - “the time factor”. SOA can be seen as organic in its nature, it’s purpose is its adaptability and ability to change. It is important to facilitate this adaptability, but the first step is to identify the possible origins of the evolution.
- The gap that exists between the architectural creation of a SOA and development of services.
The architecture and services are often developed in parallel trails. How is it possible to be sure that the developed services comply with an incomplete architecture?

Generally I see the new and the old thesis problems as covering the same issues, but using the knowledge I have gained throughout the process I have chosen to rephrase it in order to make it more precise.

The most important change is that I not explicitly mentioned EA. But what was originally written was that SOA had lifted IT to the managerial level, which I were I now reference to EA. I have also removed “The time factor”. This is however still a very big part of the thesis; both in the motivation and the content.



14 Bibliography

Bibliography is organized using the Vancouver system

- [1] Scott A. Bernard, An Introduction to Enterprise Architecture, Author House, September 2004, ISBN 1418498084
- [2] Martin Keen, Amit Acharya, Susan Bishop, Alan Hopkins, Sven Milinski, Chris Nott, Rick Robinson, Jonathan Adams, Paul Verschueren, Patterns: Implementing an SOA Using an Enterprise Service Bus, IBM, July 2004, ISBN: 0738490008
- [3] Carol O'Rourke, Neal Fishman and Warren Selkow, Enterprise Architecture Using the Zachman Framework, Thomson, 2003, ISBN: 0-619-06446-3
- [4] Ethan Cerami, Web Services Essentials, O'Reilly, February 2002, ISBN: 0-596-00224-6
- [5] Jane Carbone, IT Architecture Toolkit, Prentice Hall, May 10, 2004, ISBN: 0131473794
- [6] Gregor Hohpe, Bobby Wolf, Enterprise Integration Patterns, Addison-Wesley, December 2003, ISBN: 0-321-20068-3
- [7] MITRE Corporation, EABOK - Guide to the (Evolving) Enterprise Architecture Body of Knowledge, MITRE Corporation, 6 February 2004.
- [8] Doug Kaye, Loosely Coupled, Rds Associates Inc; 1st edition, August 2003, ISBN: 1881378241
- [9] Jeff Scott, Collaborative Enterprise Architecture Governance, Executive Update Enterprise Architecture Advisory Service Vol. 7, No. 20.
- [10] Lawrence Wilkes, Web Services Management, Part 2 - Vendor Capabilities and User Intentions, December 2004, CBDI Journal
- [11] David Sprott, Component Based Services, December 1999, CBDI Forum
- [12] Ronald Schmelzer and Jason Bloomberg, Three roads to the SOA Implementation Framework, 07 Apr 2004 (http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci958544,00.htm)
- [13] Richard Veryard, SOA Process - Case Example, CBDi Journal 2003 / July/August
- [14] B. Iyer, R. Gottlieb, The Four-Domain Architecture: An approach to support enterprise architecture design, 2004, IBM SYSTEMS JOURNAL, VOL 43, NO 3
- [15] Yefim V. Natis, Service-Oriented Architecture Scenario, Gartner Research ID Number: AV-19-6751, 16 April 2003 (<http://www.gartner.com/resources/114300/114358/114358.pdf>)
- [16] Richard Veryard, Service Oriented Architecture Frameworks, December 2004, CBDI Forum.
- [17] Lawrence Wilkes, Time to Board the Enterprise Service Bus?, July 2004, CBDI Forum
- [18] David Sprott, SOA: An Introduction for Managers, July 2004, CBDI Forum (<http://www-1.ibm.com/services/us/bcs/pdf/soa-cbdi-report-2004-july.pdf>)



- [19] Actional Corporation, SOA Command and Control, The Next Generation of SOA Enablement, October 2004, Actional Corporation
- [20] Oliver Sims, Services Oriented Architecture - Part 3 - Federation, May 2003, CBDIJournal
- [21] David Sprott and Lawrence Wilkes, Understanding SOA, September 2003, CBDIJournal
- [22] Chris Haddad, Where's the ROI?, February 10, 2005 (http://www.ftponline.com/weblogicpro/2005_03/magazine/columns/soapbox/)
- [23] Philip Allega, EA and SOA: Balancing Project Implementation and Top Down Guidance, January 2005, EACommunity.com (<http://www.eacommunity.com/articles/openarticle.asp?ID=2060>) [Read: Thursday, 24 February 2005]
- [24] Sun Microsystems, Inc., Assessing Your SOA Readiness, June 2004
- [25] Authors: David Trowbridge, Ward Cunningham, Matt Evans, Larry Brader, Paul Slater, Describing the Enterprise Architectural Space, June 2004, Microsoft Corporation (<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/entarch.asp>)
- [26] Olaf Zimmermann, Pal Krogdahl, Clive Gee, Elements of Service-Oriented Analysis and Design, 2 Jun 2004 (<http://www-106.ibm.com/developerworks/library/ws-soa-design1/>) [Read Monday, January 24th, 2005]
- [27] Ali Arsanjani, Service-oriented modelling and architecture, 12/11/2004 8:29 (<http://www-106.ibm.com/developerworks/library/ws-soa-design1/>) [Read: Monday, January 24th, 2005]
- [28] John Seely Brown, Scott Durchslag, and John Hagel III, Loosening up: How process networks unlock the power of specialization (http://www.mckinseyquarterly.com/ab_g.aspx?ar=1202&L2=1&L3=24)
- [29] John Zachman, Zachman Framework for Enterprise Architecture (<http://zifa.com/>)
- [30] Tarak Modi, ESB and SOA: The Even Couple, 08 April 2005 (<http://www.webservicespipeline.com/trends/160503220>) [Read: 22 April 2005]
- [31] Mikkel Hippe Brun, Kort introduktion til XML og Web Services, 04/03-2003 (<http://www.oio.dk/XML/publikationer>) [Read: 20 April 2005]
- [32] Alice LaPlante, Sifting Through The Alphabet Soup Of Web Services Acronyms, April 13, 2005 (<http://www.webservicespipeline.com/160702326>) [Read: 18/04/2005]
- [33] Jason Bloomberg, Calling the Elusive Enterprise Architect: You're More Important than Ever (<http://www.zapthink.com/report.html?id=ZAPFLASH-09182003>) [Read: 07 April 2005]
- [34] searchwebservices staff report, Revisiting the definitive SOA definition, 12 Jan 2005 (http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci1044083,00.html) [Read: 11 April 2005]
- [35] Blog reference
Blog name: ZDNET
Blog title: Between the Lines
Post title: ROI, step by step



Url: <http://blogs.zdnet.com/service-oriented/?p=256> [Read: 4/15/2005]

[36] Blog reference

Blog name: ZDNET

Blog title: Between the Lines

Post title: Keep SOA projects in check with top-down guidance

Url: <http://blogs.zdnet.com/BTL/?p=719> [Read: 18 April 2005]

[37] Blog reference

Blog name: Wolf's Witterings

Blog title: Mike "Wolf" Gilbert expresses his views on Enterprise Architecture

Post title: A Comparison of Service Oriented Architecture (SOA) and Event Driven Architecture (EDA)

Url: <http://blogs.msdn.com/mikegil/archive/2005/01/21/357953.aspx> [Read: 23 March 2005]

[38] Richard Veryard, Business-Driven SOA 2, June 2004, CBDI Journal p. 25-31

[39] Mark Colan, SOA - So What? 2004, IBM

[40] Lawrence Wilkes, The Essential Guide To Service Orientation, 2004, CBDI Forum

[41] CBDI Forum members, Communicating SOA, 2005, CBDI Forum

[42] David Sprott, Moving to SOA (<http://roadmap.cbdiforum.com/reports/soa/index.php>) [Read: 15 March 2005]

[43] Whit Andrews, Be Vigilant When Adopting Web Services Standards, 23 November 2004, Gartner G00123999

[44] T. Bittman, Gartner Introduces the Infrastructure Maturity Model, 19 November 2004, Gartner Research Note

[45] MEGA, Challenges and Methods for the Implementation of SOA, February 2004 (http://www.mega.com/en/download/brochure_download/pdf/mega_archi_service.pdf)

[46] David Sprott and Lawrence Wilkes, Enterprise Framework for SOA, March 2005, CBDI Journal

[47] Priser, (<http://cvr.dk/cvr-online.html>) [Read: 01 April 2005]

[48] Web Services Glossary (<http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>) [Read: 01 February 2005]

[49] Blog reference

Blog name: johnhagel.com

Blog title: Perspectives

Post title: Loosely Coupled: A Term Worth Understanding

Read: 8/3-2005

Url: <http://www.johnhagel.com/blog20021009.html>

[50] looselycoupled.com (<http://looselycoupled.com/glossary/loose%20coupling>)

[51] Dr. Hao, Thomson Corporation (<http://lists.w3.org/Archives/Public/www-ws-arch/2003Jan/0341.html>)



- [52] creator of Business Process Management Language (BPML)
(<http://lists.w3.org/Archives/Public/www-ws-arch/2003Jan/0341.html>)
- [53] hyperdictionary.com (<http://hyperdictionary.com/>)
- [54] Ronald Schmelzer, The ROI of SOA, 27 January 2005
(<http://www.zapthink.com/report.html?id=ZAPFLASH-20050127>) [Read: 31 Marts 2005]
- [55] Lamont Wood, The Costs and Benefits of SOA, 9. august 2004
(<http://www.managingautomation.com/maonline/magazine/read.aspx?id=1245185>) [Read: 31 Marts 2005]
- [56] Jack Greenfield, The Case for Software Factories, August 24, 2004
(<http://www.itarchitect.co.uk/enterprise/display.asp?id=96>)
- [57] Peter Herzum, Applying Enterprise Architecture, 2003, Executive Report, Vol. 6, No. 3
- [58] Clayton M. Christensen, The Innovator's Dilemma, Harvard Business School Press, June 1997, ISBN: 0875845851
- [59] Henrik Zangenberg and Christer Forsberg, The SOA Tutorial, April 29 2005, Gartner
- [60] On Demand Business (<http://www-306.ibm.com/e-business/ondemand/us/index.html>)
[26 May 2005]
- [61] Using the Business Process Execution Language (BPEL) to Integrate Business Processes, March 2005 (<http://news.pghtech.org/teq/teqstory.cfm?ID=1329>) [Read: 26 May 2005]
- [62] David Sprott, SOA IS A BUSINESS ISSUE, 25th May 2005, CBDI
(<http://www.cbdiforum.com/public/news/index.php3?id=1474>)
- [63] Bernhard Borges, Kerrie Holley and Ali Arsanjani, Delving into Service-Oriented Architecture (<http://www.developer.com/design/article.php/3409221>) [read: 18 May 2005]
- [64] Richard Veryard, Practical Support for Separate Supplier and Consumer Activity, CBDI, 2003 (<http://roadmap.cbdiforum.com/reports/select/index.php>)
- [65] J. A. Zachman, A framework for information systems architecture, 1987, International Business Machines Corporation
- [66] James McGovern, Scott Ambler, Mike Stevens, James Linn, Vikas Sharan, Elias K. Jo, A Practical Guide to Enterprise Architecture, Prentice Hall PTR , December 2003, ISBN: 0131412752
- [67] Scott Ambler, Agile Enterprise Architecture: Beyond Enterprise Data Modeling (http://www.flashline.com/Content/Ambler/agile_ent_arch.jsp?sid=1116166268313-1123631162-101#Why) [Read: 01 June 2005]
- [68] Gurpreet S. Pall, Microsoft Architectural Frameworks, 2005, Microsoft Corporation
(http://download.microsoft.com/documents/australia/MSDN/RAF_2005/Gurpreet-Microsoft_Architecture_Framework-2.ppt)
- [69] The Red Queen Principle (<http://pespmc1.vub.ac.be/REDQUEEN.html>) [Read: 01 June 2005]



[70] META Group Research - Trend, META Group Research, 3 February 2005, META Group Inc. (<http://www.metagroup.com/us/displayArticle.do?fileName=trend2056>) [Read: 02 June 2005]

[71] Dan Ruby, Build a Solid Foundation for Agile IT, June 9, 2004, FTPOnline (<http://www.ftponline.com/ea/magazine/summer2004/features/druby/>)

[72] Signe Wagner, Den Serviceorienterede Forvaltning, Juni 2005 (http://signewagner.dk/Speciale/Samlet_v.1.pdf)

[73] Alan Perkins, Implementing the Zachman Framework for Enterprise Architecture, 1997, Visible Systems Corporation (<http://www.ies.aust.com/~visible/papers/Zachman.html>)

[74] Blog reference

Blog name: Harry Pierson's DevHawk Weblog

Blog title: Passion * Technology * Ruthless Competence

Post title: Gurpreet on Enterprise Architecture

Url: <http://blogs.msdn.com/devhawk/archive/2004/09/15/229850.aspx#comments> [Read: 01 June 2005]

[75] Geoffrey A. Moore, Inside the Tornado, HarperBusiness, December 2004, ISBN: 0060745819

[76] Wikipedia, people from around the world (<http://en.wikipedia.org>) [Read: 06 June 2005]

[77] Nicholas G. Carr, IT Doesn't Matter , May 1, 2003, Harvard Business School Press, ISBN: B00009MBYN

[78] Transformative Solutions for any Business, New Information Paradigms (NIP) (<http://www.nipltd.com/Approach/Methodologies/Chasm.html>) [Read: 06 June 2005]

[79] Blog reference

Blog name: Service Oriented Enterprise

Author: Jeff Schneider

Post title: SOA - PHASE III

Url: http://schneider.blogspot.com/archives/2005_05_01_schneider_archive.html#111506045486190107 [Read: 06 June 2005]

[80] Rich Salz, Introducing WS-I and the Basic Profile, March 04 2003 (<http://webservices.xml.com/pub/a/ws/2003/03/04/endpoints.html>)

[81] Luis Felipe Cabrera, Christopher Kurt, Don Box, An Introduction to the Web Services Architecture and Its Specifications, October 2004, Version 2.0 (http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnwebsrv/html/introwsa.asp#intr_topic37)

[82] Lawrence Wilkes, The Web Services Protocol Stack, Feb 2005, CBDI (<http://roadmap.cbdiforum.com/reports/protocols/>)

[83] David Sprott, A Web Services Maturity Model

(<http://roadmap.cbdiforum.com/reports/maturity/>) [Read: 07 July 2005]

[84] Kunal Mittal, Build your SOA: Maturity and methodology, Part 1, 20 May 2005

(<http://www-128.ibm.com/developerworks/webservices/library/ws-soa-method1.html>)



- [85] Kunal Mittal, Service Oriented Architecture Maturity Model (SOAMM) (<http://www.soaconsultant.com/html/soamm.shtml>) [Read: 07 July 2005]
- [86] David S. Linthicum, ESB versus Fabric...Stop It!, Sunday 22 May 2005 (<http://www.webservices.org/index.php/ws/content/view/full/63539>)
- [87] Jean-Jacques Dubray, Fundamentals of Service Orientation, February, 2005 (http://www.attachmate.com/NR/rdonlyres/9AB80055-9428-4453-9F7C-A1A2A7C32FAE/0/tp_ssb_SOA.pdf)
- [88] Kumar, K. et Al., ERP Experiences and Evolution, Communications of the ACM, April [2000]/vol. 43. No. 4 page 23-26
- [89] Boris Lublinsky, Dmitry Tyomkin, Dissecting Service-Oriented Architectures, October 2003, Business Integration Journal (<http://bijonline.com/PDF/Oct03Lublinsky.pdf>)
- [90] California Enterprise Architecture Defined, Federal CIO Council of California (http://www.cio.ca.gov/ITCouncil/Committees/PDFs/Ca_EA_Defined.pdf) [Read: 21 July 2005]
- [91] Don Free, Maria Kun, Architectural Influences on Core Banking Replacement, 2005, Gartner
- [92] Infoconomy Staff, Software at your service, 19 April 2005 (<http://www.infoconomy.com/pages/xmlwebservices-analysis/group105878.adp>)
- [93] Sue Bushell, ATO Eyes SOA, 20/07/2004 (<http://www.cio.com.au/index.php/id;1832424547;fp;4;fpid;21>)
- [94] John A. Zachman, Enterprise Architecture Artifacts Vs. Application Development Artifacts, 2005 (http://www.tdan.com/ZIFA_05.pdf)
- [95] Carla Marques Pereira and Pedro Sousa, A Method to Define an Enterprise Architecture using the Zachman Framework, 2004, ACM Symposium on Applied Computing (<http://delivery.acm.org/10.1145/970000/968175/p1366-marques.pdf?key1=968175&key2=8520984211&coll=GUIDE&dl=GUIDE&CFID=51371281&CFTOKEN=5866181>)
- [96] Blog reference
Blog name: Service Oriented Enterprise
Author: Jeff Schneider
Post title: CIO's Aren't Taking SOA Serious
Posted: Sunday, June 12, 2005
Url:http://schneider.blogspot.com/archives/2005_06_12_schneider_archive.html#111858349414143623 [Read: 06 June 2005]
- [97] Jeffrey Kaplan, Strategic IT Portfolio Management, PRTM Inc., July 2005, ISBN: 0976609304
- [98] David C. Hay, Requirements Analysis: From Business Views to Architecture, Prentice Hall PTR; 1st edition, August 2002, ISBN: 0130282286
- [99] Infostructurebase, The Danish Ministry of Science, Technology and Innovation (<http://isb.oio.dk/info/>) [Read: 22 July 2005]



- [100] Ed Scannell, IBM delivers SOA enablers, April 21, 2004
(http://www.infoworld.com/article/04/04/21/HNibmsoas_1.html) [Read: 29 August 2005]
- [101]
Title: Danish Bank Uses Visual Studio .NET, Web Services to Generate New Revenue Sources, Microsoft Corporation, February 27, 2003
(<http://www.microsoft.com/resources/casestudies/CaseStudy.asp?CaseStudyID=13756>)
- [102] David Sprott, SOA IS A BUSINESS ISSUE, 25th May 2005
(<http://www.cbdiforum.com/public/news/index.php3?id=1474>)
- [103] David Sprott, The SOA Governance Framework, September 2004
(http://www.cbdiforum.com/secure/interact/2004-09/soa_governance_framework.php)
- [104] Richard Veryard, Business-Driven SOA CBDi Journal 2004 / May
(http://www.cbdiforum.com/secure/interact/2004-05/Business_Driven_SOA.php)
- [105] Richard Veryard, Orchestration Patterns Using BPEL, CBDi Journal 2004 / October
(http://www.cbdiforum.com/secure/interact/2004-10/orchestration_patterns_bpel.php)
- [106] Ronald Schmelzer, Outsourcing, SOA, and the Industrialization of IT, ZAPFLASH - 10132004 (<http://www.zaphink.com/report.html?id=ZAPFLASH-10132004>)
- [107] Ben Braue and Sean Kline, SOA governance: a key ingredient of the Adaptive Enterprise, February 2005 (http://devresource.hp.com/drc/resources/soa_gov/index.jsp)
- [108] Judith Myerson, Use SLAs in a Web services context, Part 1: Guarantee your Web service with a SLA, 01 Apr 2002 (updated 29 Oct 2004) (<http://www-128.ibm.com/developerworks/library/ws-sla/>)
- [109] Richard Veryard, Business-Driven SOA 2, CBDi Journal 2004 / June
(http://www.cbdiforum.com/secure/interact/2004-06/Business_Driven_SOA2.php)
- [110] Scott A. Bernard, An Introduction to Enterprise Architecture - 2nd edition, Not published yet.
- [111] Lawrence Wilkes, COMMENTARY: TO ESB OR NOT TO ESB?, 2nd September 2005
(<http://www.cbdiforum.com/public/news/index.php3?id=1485>)
- [112] Understanding SOA, ESB and Web Services, Cape Clear
(<http://www.capeclear.com/technology/index.shtml>) [Read: 13 September 2005]
- [113] Microsoft, Microsoft on the Enterprise Service Bus (ESB), July 2005, Microsoft Corporation (<http://home.comcast.net/~sdwoodgate/MicrosoftonESB.doc>)
- [114] Michael Sparling, Lessons Learned through Six Years of Component-based Development, 2000 ACM 0002-0782/00/1000
(<http://www.castek.com/PDF/LessonsLearned.pdf>)
- [115] Rockford Lhotka, SOA Versioning Covenant, April 20, 2005, The Serverside .NET
(<http://www.theserverside.net/articles/showarticle.tss?id=SOAVersioningCovenant>)
- [116] Blog reference
Blog name: SOA Life



Author: Jared Rodriguez

Post title: Release Management in a SOA

Posted: February 14, 2005

Read: 19 September 2005

Url: <http://www.bloglines.com/blog/JaredRodriguez?id=2>

[117] Vance McCarthy, Web Services Testing Beyond SOAP: Brace for Changes, 11/24/2002, Open Enterprise Trends (http://www.oetrends.com/cgi-bin/page_display.cgi?141)

[118] Chris Peltz; Anjali Anagol-Subbarao, Design Strategies for Web Services Versioning, Adapting to the needs of the business, Apr. 5, 2004, SOA Web Services Jurnal (<http://webservices.sys-con.com/read/44356.htm?CFID=222368&CFTOKEN=131B5FDF-12A1-1218-0EF0F30844C51ACB>)

[119] Peter M. Senge, The Fifth Discipline, Currency, January 1994, ISBN: 0385260954

[120] Michael Bean, Bullwhips and Beer: Why Supply Chain Management is so Difficult ()
[Read: 19 September 2005]

[121] Web Services Architecture Requirements, W3C Working Draft 29 April 2002 (<http://www.w3.org/TR/2002/WD-wsa-reqs-20020429#N100CB>) [Read: 26 September 2005]

[122] Henrik Hvid Jensen, Service Orienteret Arkitektur - Integration som konkurrenceparameter, Litera, 2004, ISBN: 87-91242-34-7

[123] <http://standarder.oio.dk/English/>, The Danish government, version 1.2.5 dated September 23, 2005 (<http://standarder.oio.dk/English/>) [Read: 07 October 2005]

[124] Joseph Poozhikunnel, The X-Factor in SOA, 01 April 2005 (<http://www.15seconds.com/issue/050831.htm>)

[125] David Trowbridge, Ward Cunningham, Matt Evans, Larry Brader, Paul Slater, Wadeware. Describing the Enterprise Architectural Space, June 2004, Microsoft Corporation (<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/entarch.asp>)
[Read: 8/3-2005]

[126] Jakob Burkard, Sammenhængen mellem Service Orienteret Arkitektur (SOA) og forretningsprocesser, 2005, Capgemini

[127] Robert L. Glass, V. Ramesh, Iris Vessey, An Analysis of Research in Computing Disciplines, June 2004/Vol. 47, No. 6 COMMUNICATIONS OF THE ACM (<http://portal.acm.org/citation.cfm?id=990680.990686>)

[128] Alex Cullen with Laurie M. Orlov, Samuel Bright, What Must EA Do To Sustain SOA? September 30, 2005, Forrester (<http://www.forrester.com/Research/Document/Excerpt/0,7211,37855,00.html>) [Read: 26 October 2005]

[129] Jon Bachman, Movin' SOA On Up: Introducing a New Service-Oriented Architecture Maturity Model, Sonic Software, September 19, 2005 (http://www.sonicsoftware.com/solutions/learning_center/soa_insights/movin_soa_on_up/index.ssp) [Read: 26 October 2005]

[130] Luis Cupajita & Omid Pourzanjani, June 12, 2003, Mindspeed (<http://members.cox.net/jimsutter/meeting/web-services.ppt>)