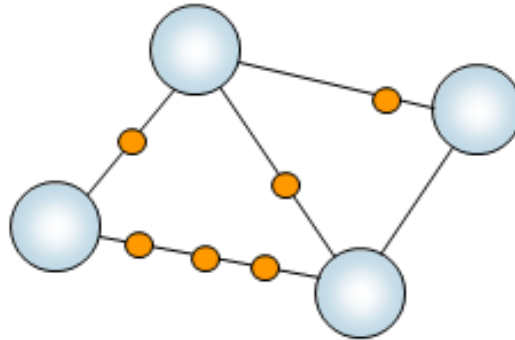




UNIVERSITY OF GOTHENBURG



Event-Driven Architecture and SOA in collaboration

**A study of how Event-Driven Architecture (EDA)
interacts and functions within Service-Oriented
Architecture (SOA)**

BEHROOZ MALEKZADEH

Master of Thesis in IT-Management

Report No. 2010:056

ISSN: 1651-4769

Summary

Over the years many different architecture styles and concepts have evolved. Two are Service Oriented architecture (SOA) and Event Driven Architecture (EDA). Both styles are revolutionary and have great benefits for the business if they are used in the right context for the right purpose. EDA is not a new paradigm nor is it new as an architecture style. It has been around for many years but has revived during the last years due to SOA and new technology available. Now days EDA is often mentioned when discussing SOA. The confusion these days is about how the two architecture styles interact. People have different views on this issue. Now, even as there is widespread concurrence that SOA brings in the possibility of EDA being used, there is a lot of debate on exactly how EDA will blend in with SOA. Ranging from EDA being the “new SOA”, to EDA “succeeding” SOA, to EDA “extending” SOA, to pure skepticism of any relationship at all!

The purpose of this thesis is to study SOA and EDA and discuss how they interact and integrate in the same environment. The discussions and analysis presented will be at a conceptual level and I will not cover technical infrastructure issues if not necessary.

This research is founded on theoretical study, personal experience, and interviews to find out the main characteristics, similarities and differences of both architecture concepts and how they are relating.

During the study it became evident that there are several areas where EDA and SOA are interacting. People have different opinion on how EDA and SOA relate to each other but I’m of the clear opinion that EDA is extending SOA in several areas. The relation between the two concepts is obvious almost in all architecture layers and aspects, from business and information, to information system integration and technical infrastructure. It is however important to also point out that despite the identified relations between EDA and SOA they may be implemented separately.

Based on our findings there are three main reasons for this collaboration. The first one is common and aligned business objectives, the second one is that both architecture concepts build upon decoupled and flexible components and common data model, and the third reason is use of common infrastructure and technology. Though the relation between EDA and SOA is clear the challenges when implementing EDA and SOA should not be underestimated. Involving business when implementing EDA and SOA is the key to success and perhaps the main challenge. Another major challenge is the governance of services and events. This is also a new field where the level of maturity may not be high enough and where real-life experience is rare.

Keywords: Architecture, Enterprise Architecture, Service-Oriented Architecture, Event-Driven Architecture, Event, Service

Supervisor: Kalevi Pessi

Acknowledgements

I would like to thank Jan Mattsson, Lennart Eriksson, and finally Joshua Oyugi for their valuable input to this research. Furthermore I want to show appreciation to my tutor Kalevi Pessi at IT University of Göteborg who advised and helped me shape the content and structure of this research. Thank you!

Behrooz Malekzadeh
May 2010,
Göteborg.

Table of Contents

1. INTRODUCTION	7
1.1 BACKGROUND	7
1.2 PURPOSE.....	8
1.3 QUESTION OF ISSUE.....	8
1.4 STRUCTURE OF THESIS	8
2. METHODOLOGY	9
2.1 RESEARCH METHOD.....	9
2.1.1 <i>Positivism and hermeneutic</i>	9
2.1.2 <i>Explanatory and descriptive research</i>	10
2.1.3 <i>Quantitative and qualitative research methodology</i>	11
2.2 DATA COLLECTION METHOD.....	11
2.2.1 <i>Secondary sources</i>	11
2.2.2 <i>Primary sources</i>	11
2.3 RELIABILITY AND VALIDITY	13
2.2 APPLIED RESEARCH STRATEGY.....	13
3. THEORETICAL BACKGROUND	15
3.1 IS/IT MANAGEMENT	15
3.2 ARCHITECTURE	17
3.3 ENTERPRISE ARCHITECTURE	18
3.3.1 <i>Enterprise Architecture Frameworks</i>	19
3.3.2 <i>Business architecture</i>	21
3.3.3 <i>Information architecture</i>	22
3.3.4 <i>Information System architecture</i>	22
3.3.5 <i>Technology Infrastructure architecture</i>	24
3.4 TWO ARCHITECTURE CONCEPTS	24
3.5 SUMMARY	25
4 SERVICE-ORIENTED ARCHITECTURE (SOA)	26
4.1 BACKGROUND	26
4.2 CHARACTERISTICS OF SOA	27
4.3 CONCEPT	27
4.4 PRINCIPLES.....	30
4.5 DATA.....	33
4.6 SOA IMPLEMENTATION COMPONENTS.....	34
4.7 ENTERPRISE SERVICE BUS (ESB)	35
4.8 BUSINESS PROCESS MANAGEMENT (BPM) AND BUSINESS ACTIVITY MONITORING (BAM).....	36
4.9 WEB SERVICES	37
4.10 SUMMARY	38
5 EVENT-DRIVEN ARCHITECTURE (EDA)	39
5.1 BACKGROUND	39
5.2 CHARACTERISTICS OF EDA.....	39
5.3 CONCEPT	40
5.4 PRINCIPLES.....	41
5.5 DATA.....	42
5.6 EDA IMPLEMENTATION COMPONENTS	43
5.7 COMPLEX EVENT PROCESSING (CEP)	44
5.8 SUMMARY	45
6. ANALYSIS	45
6.1 COMPARATIVE ANALYSIS	45
6.1.1 <i>View on Business</i>	45
6.1.2 <i>View on Information</i>	46
6.1.3 <i>View on Information System</i>	48

6.1.4 View on Integration	48
7 DISCUSSION.....	50
7.1 BUSINESS OBJECTIVES	50
7.2 ENTERPRISE ARCHITECTURE CONTEXT.....	51
7.3 ARCHITECTURAL RELATIONS AND DEPENDENCIES	52
7.4 SERVICE DECOUPLING	54
7.5 REAL-TIME INFORMATION SHARING	56
7.6 COMMON COMPONENTS DURING IMPLEMENTATION	57
7.7 MAIN CHALLENGES OF COMBINING SOA AND EDA	57
8. CONCLUSIONS.....	59
8.1 INTERACTION BETWEEN SOA AND EDA	59
8.2 PROPOSALS FOR FUTURE RESEARCH	60
LIST OF REFERENCES	62
APPENDIX A – ZACHMAN ENTERPRISE ARCHITECTURE FRAMEWORK	66
APPENDIX B – SOA IMPLEMENTATION COMPONENTS.....	67
APPENDIX C – EDA IMPLEMENTATION COMPONENTS	68
APPENDIX D – INTERVIEWS	69

Table of Figures

Figure 1 Positivistic and Hermeneutic approach (Patel and Davidson 1994)	10
Figure 2 Research Process	14
Figure 3 Strategic alignment	15
Figure 4 Formal Links (Whittle and Myrick 2005).....	19
Figure 5 Zachman Enterprise Architecture Framework (Zachman 1996:7).....	20
Figure 6 Unified Architecture Framework (Maes et al. 2000:19).....	21
Figure 7 Request/reply mechanism in a SOA	28
Figure 8 Concept of Service Orientation.....	28
Figure 9 Elements of SOA (Marks and Bell 2006:6).....	29
Figure 10 Coarse-Grained Services.....	32
Figure 11 ESB architecture (Khoshafian 2007)	36
Figure 12 Different IT capabilities enabling BPM (Campbell and Mohun 2007)	37
Figure 13 The publish/subscribe mechanism in an Event-Driven Architecture	40
Figure 14 CEP function within EDA	44
Figure 15 Comparative Analysis – Business process and function.....	46
Figure 16 Comparative Analysis – Information.....	47
Figure 17 Comparative Analysis - Information System.....	48
Figure 18 Comparative Analysis - Integration	49
Figure 19 Service and Event relation	53
Figure 20 SOA and EDA Architecture Layers.....	53
Figure 21 SOA Business Process	54
Figure 22 SOA and EDA Interaction	54
Figure 23 SOA and EDA Architecture.....	55
Figure 24 SOA & EDA information handling	56
Figure 25 ESB supporting Event Management.....	57
Figure 26 Summary of how EDA and SOA interacts	60
Figure 27 Zachman Enterprise Architecture Framework.....	66
Figure 28 IBM On Demand Operation Environment (ODOE).....	67
Figure 29 Major implementation components within EDA (Michelson 2006)	68
Figure 30 SOA and EDA Architecture Layers.....	71

1. Introduction

This chapter is an introduction to the topic and gives a discussion of the problem area of interest, which will be narrowed down to the purpose of the thesis, the question of issue, and the methodology.

1.1 Background

Today most enterprises and organizations in all sectors are highly dependent on their information systems. Information technology has become unavoidably aligned with business. With the emergence of e-commerce the use of technology is becoming an obvious way of doing business. Consequently organizations are increasingly looking toward using new technology not only to intensify existing operations but also to create new opportunities and new competitive advantages. In order to manage information systems and information technology strategically, it is helpful to understand how the role of information systems has evolved in the organization. Many organizations would like to rethink their IT investments, but unfortunately have a legacy resulting from a less than strategic approach in the past. (Ward and Peppard, 2002)

Both business and technology has developed in parallel over the years however there has always been a gap separating the two entities. Aerts et al (2003) also agree that the ability of enterprises to be competitive is mostly dependent on information systems and technology platforms. But perhaps most importantly how business and technology are aligned and focusing on reaching same business objectives.

The role of information system and IT has been growing to become a strategic part of the business. IT-management, IT-strategy and Enterprise Architecture are becoming obvious parts of the daily operation in enterprises. Ward and Peppard (2002) define IT strategy as a definition of organization's demand for information and systems to support the overall strategy. According to Zachman (1996) Enterprise Architecture is the cornerstone for leveraging technology innovations to fulfill the expectations of a viable and dynamic information age enterprise.

Over the years many different architecture styles and concepts have evolved. Two are Service Oriented architecture (SOA) and Event Driven Architecture (EDA). Both styles are revolutionary in how they try to link business with IT and have great benefits for the business if they are used in the right occasion for the right purpose. EDA is not a new paradigm nor is it new as an architecture style. It has been around for many years but has revived during the last years due to SOA and the new technology available. Now days EDA is often mentioned when discussing SOA and the content of it. A survey done by Gartner in June 2008 reveals the fact that of all organizations building SOA, 37% are leveraging EDA in some aspect and that the number will rise to 54% during the next 12 months. (Schulte and Sholler 2008). The confusion these days is about how the two architecture styles interact. People have different views on this issue. Now, even as there is widespread consensus that SOA brings in the possibility of EDA being used, there is a lot of debate on exactly how EDA will blend in with SOA. Ranging from EDA being the "new SOA", to EDA "succeeding" SOA, to EDA "extending" SOA, to pure skepticism of any relationship at all! Some say that EDA is usually implemented as a type of SOA, stressing the use of fully asynchronous, one-way communication patterns, rather than the more common client/server SOA communication patterns, such as request/reply! What most people agree upon is however that EDA and SOA

must collaborate and work together in the same environment. To really extract EDA benefits, SOA is a necessity. SOA infrastructure, services, principles, and architecture style will help you implement your event driven architecture.

1.2 Purpose

As described in the background there are different opinions about SOA and EDA and the way they are functioning together. The purpose and goal of this thesis is basically to clarify if and how SOA and EDA can or should function in the same environment to reach promised benefits by both styles. The study seeks to contribute to this discussion by pursuing two specific goals:

1. The study aims to address and discuss similarities and differences of SOA and EDA and get a better understanding of how SOA and EDA complement or differentiate
2. The study aims to identify interaction points and characteristics of an architecture where both EDA and SOA are applied to reach promised benefits

The discussions and analysis presented will be at a conceptual level and it will not cover technical or infrastructure issues if not necessary.

1.3 Question of Issue

The question of issue in this thesis is:

How can Event Driven Architecture interact and function with Service Oriented Architecture?

To be able to answer the question two research questions are formulated. The first one is discussing and explaining SOA and EDA. This leads to the first research question:

1. What are the main characteristics of SOA and EDA?

The first question will generate necessary information to conduct a comparative analysis between the two architecture styles. This leads to research question two:

2. What are the main similarities and differences when comparing EDA and SOA?

The second question provides necessary material to be able to analyze and discuss the issue from different aspects to be able to answer the question of issue.

1.4 Structure of Thesis

Chapter one – is describing the background and the problem area of interest and later on narrowed down to the question of issue.

Chapter two – is describing the methodology and process to conduct this study.

Chapter three – is giving the reader a theoretical background which is needed to understand the discussions in the following chapters. This chapter covers definitions such as IS/IT management, architecture, and Enterprise Architecture.

Chapter four – outlines the main characteristics and principles of Service Oriented Architecture. Main concepts within SOA are described in detail.

Chapter five – outlines the main characteristics and principles of Event-Driven Architecture. Main concepts within SOA are described in detail.

Chapter six – analyzes the data collected and discuss the outcome of the thesis based on a comparison between theoretical data, established theories, and frameworks.

Chapter seven – summarizes the analysis in the previous chapter and illustrates the interactions points between SOA and EDA and how they collaborate in an architecture context.

Chapter eight – presents a conclusion answering the purpose of the question of issue.

2. Methodology

The applied scientific research approach is motivated and described within this section. The research process is explained in detail in order to understand how the data has been collected and analyzed.

2.1 Research method

According to Patel and Davidson (1994) all scientific research have some common characteristics. They are all aiming at producing new knowledge and adding new dimensions to existing theories. They are all based upon a solid theoretical base leveraging established theories and models. And the research must fulfill some predefined scientific requirements and align with existing rules and approaches. Therefore it is very important to know how to conduct research to attain acceptable results. In the following section we will go through several research approaches beginning with a comparison between positivistic and hermeneutic approach.

2.1.1 Positivism and hermeneutic

Two traditional scientific approaches are positivistic and hermeneutic which are different in the aspect of their methodology. Positivism is a scientific approach having its roots in empirical studies and is used by scientists who study a research question from an external point of view. The positivistic approach is paying attention to knowledge achieved through measurement and objective identification. Another characteristic of positivism is the idea of breaking down the research topic into several parts which are study one by one. It is also of major importance that researcher's personal view and opinion are not allowed to affect the result. (Patel and Davidson 1994).

Hermeneutics is the opposite of positivism. Hermeneutic approach explains relationships by a more personal interpretative development. This approach was mainly used during 17th and 18th century to interpret religious manuscripts and has since that been a central approach when dealing with anthropology. In this approach the researcher is approaching the topic in a more subjective matter allowing use of personal experience and feelings. In opposition to

positivism trying to analyze the subject piece by piece hermeneutic is focusing on the general impression, holistic view, and generalization. (Patel and Davidson 1994).

	Positivistic	Hermeneutic
Research aim	Research concentrates on description and explanation.	Research concentrates on understanding and interpretation.
Scope	Well-defined, measurable occurrences, most often represented in natural science e.g. physics.	Holistic view, generalization, based upon personal experience most often represented in cultural- and human science.
Researcher's position	Logical, analytical, and objective. The researcher maintains a distance between themselves and subject of research.	Personal feelings and experience are used. Personal involvement.
Methodology	Breaking down the problem area into smaller areas (deductive), empirical.	Understanding and interpreting.
Data	Statistical and mathematical techniques for quantitative processing of data is central.	Primarily non-quantitative data.

Figure 1 Positivistic and Hermeneutic approach (Patel and Davidson 1994)

The table is not only a comparison between positivism and hermeneutic approach but also the method used in this thesis. In this thesis both approaches have been used. It is very difficult to differentiate strictly between both approaches. This study has mainly applied the hermeneutic approach. The theoretical background has been focusing on understanding and interpreting existing research and literature. The analysis and conclusion parts were characterized by personal interpretation and understanding of the subject matter. As a conclusion this research is in favor of the hermeneutic approach with some positivistic elements, where some problem areas have been broken down into smaller areas and analyzed in more detail.

2.1.2 Explanatory and descriptive research

As described by Patel and Davidson (1994) there are different methods for research and studies which can be applied. Two of the main methods are *explanatory* and *descriptive* research. Explanatory research is research conducted in order to explain any behavior. Usually this approach is applied when we have limited knowledge or lack of information. The explanatory research will help us to collect new information and identify the reasons. We try to explain and not just reporting. The explanatory research is most often comprehensive considering almost all aspects of the topic to be explained.

The main goal of Descriptive type of research is to describe the topic and characteristics about what is being studied. This method is often used when there are an extensive amount of data and research already available. The idea behind this type of research is to study e.g. frequencies and models by analyzing existing data. Although this research is highly accurate, it does not gather the causes behind a situation. Descriptive research is mainly done when a researcher wants to gain a better understanding of a topic and has to carry out research in order to gain a better understanding. In most cases when applying the descriptive approach the researcher is focusing on a limited number of aspects of the topic. The selected aspects are then studied and described in detail. The description may explain each aspect separately or the interaction between them.

Descriptive research approach is the one selected and applied in this study. The choice of the descriptive approach has been obvious from the beginning due to several reasons. The most apparent reason is the definition of the question of issue and the goal with this thesis. The study is trying to describe two topics SOA and EDA from a limited number of aspects (characteristics, concept, principles, data, and implementation components). Except these aspects which are studied both within SOA and EDA other aspects have been included that are only relevant for SOA or EDA. These aspects are studied and analyzed by using existing information. The objective is to get a better understanding of both SOA and EDA and the relations between them.

2.1.3 Quantitative and qualitative research methodology

Patel and Davidson (1994) are also pointing out two other aspects of a research methodology which is *qualitative* or *quantitative*. Qualitative approach involves analysis of data such as words, pictures, or other objects which usually are gathered through interviews, literature or other artifacts. The aim with this approach is to define a complete and detailed description of the topic. The quantitative approach involves analysis of numerical data and the aim is to classify features, count them, and construct statistical models in an attempt to explain what is observed. Patel and Davidson (1994) are however of the opinion that these two approaches are in most cases combined and researchers are using both in parallel.

Data used within this study is of qualitative character. Due to the nature and content of the topic no quantitative data has been collected or analyzed. The qualitative data has been gathered mainly through literature but also interviews. The methods for collecting data are described in the following section.

2.2 Data collection method

Data collection is an important element in every study. In the section different approaches of collecting data are presented. Basically there are two different sources of data: secondary and primary data sources. Already existing data is referred to as secondary data, such as books, magazines, internet sources, and publications. Secondary source of information is the second-hand information about the happenings. Primary data is data observed and collected from first-hand sources in various ways, such as interviews, surveys, or questionnaires. (Bryman and Bell 2007).

2.2.1 Secondary sources

The data used in this study has originated mainly from secondary sources of data. Secondary sources have been used in defining the theoretical background and a conceptual framework. The study was initiated by the process of reviewing literature from previous research. The aim of the theoretical background was to understand and describe the topic field. In this category of data the author searched for information in databases at Chalmers (<http://chans.lib.chalmers.se/search/>) and Gothenburg University (<http://www.ub.gu.se/>), literature, academic publications, internet sites, and news articles. Almost all of the e-books were found by accessing the database Books24x7.

2.2.2 Primary sources

During the study as it appeared that “EDA principles” and “EDA – SOA challenges” are not fully covered by reliable secondary sources, it was necessary to use primary sources to complement and validate existing material.

In the case of interviews they can be carried out in several ways: structured, semi-structured, or unstructured. (Bryman and Bell 2007). The main method used to collect primary data in this research has been semi-structured interviews. Applying semi-structure interviews allowed us to ask specific questions though leaving some space for open discussions and analysis be the respondent. The semi-structured interviews did also allow us to collect much more data from the interviews.

When collecting the primary data with the help of the selected respondents, a qualitative approach was adopted. According to Patel and Davidson (1994) a qualitative method enables a deeper and more complete understanding of the research area and its complex nature in contrast to a quantitative method. Since personal interviews allow a very high level of interaction, this form of qualitative method was strived after. In this study three interviews were conducted. According to Davidson and Patel (1994) the reliability of interviews is increased by having trained respondents. When selecting the respondents the author has been trying to identify trained respondents according to two selection criteria.

1. The respondent must have long experience in this field either practical or theoretical
2. The respondent must have been participating and discussing the topic in other public events or forums

By searching through old IT publications e.g. ComputerSweden, forums e.g. Dataföreningens EA nätverk, and events discussing architecture trained respondents with relevant experience and competency were identified. In total three interviews where done including two telephone interviews. Due to the nature of the questions to be asked, the interview questions were sent to each respondent before the interview. That way, each respondent got the possibility to prepare for the interview, and perhaps do some research. The following respondents where finally selected.

Lennar Eriksson is an experienced and well known system architect with many years of SOA and EDA experience. He has been participating in several interviews and was selected top ten developer by Computer Sweden 2008.

Jan Mattsson is an experienced IT architect which also was selected by Computer Sweden as a top developer during 2008. He is a certified IT architect and Microsoft application developer. Jan Mattsson has also been interviewed in publications discussing both SOA and EDA.

Joshua Oyougi is an experienced SOA, EDA and integration architect. He possesses deep technical and conceptual knowledge with a solid theoretical background within IT management.

The duration of the telephone interviews where designed to be aligned with the proposed model by Esaiasson and Gilljam (2003). According to Esaiasson and Gilljam (2003) telephone interviews can be the second best choice when respondents are located on remote locations. They also point out that in case of telephone interviews the duration of the interview shall not exceed thirty minutes and number of questions must be limited. The duration of telephone interviews was limited to between 30 and 45 minutes and the number of questions was limited to three:

1. What are the main interaction points between EDA and SOA?
2. What are the main challenges when combining EDA and SOA?
3. What are the main architecture principles when designing EDA?

The purpose of these interviews is not to conduct an empirical study. The interviews are only used as another primary source for collecting data where a lack of information in literature has been obvious. The output of all interviews has been valued at the same level.

2.3 Reliability and validity

Data collection and data gathering is something we are doing on daily basis. In some cases we are getting information from sources that are already validated and approved by others. Most often these sources are used broadly and generally by many collectors. However when we construct the tools to collect data we are not sure if we get the right information. This challenge must be dealt with in two ways according to Davidson and Patel (1994). Firstly we must be sure that we are investigating what we aim to (validity) and secondly we must be sure that we are doing the research in a reliable way (reliability). Davidson and Patel (1994) are also emphasizing that reliability and validity are not excluding each other and must both be considered separately.

According to Davidson and Patel (1994) validity can be achieved in two ways. The first one is validation of the content by letting an external partner review the content and analyze the validity of the topics and the content. This is the only validation method used within this research where the validation has been performed by my supervisor at IT University in Göteborg. Both the topics and content has been analyzed and discussed.

The second approach when verifying the validity is by conducting a parallel validation where the tool developed is used in other but similar circumstances. In most occasions it is about using different techniques when studying the same topic. (Davidson and Patel 1994). This approach has also been implemented within this research by studying different architecture frameworks. The different theoretical frameworks used have been a very good base for validating the context and content of this thesis.

Reliability is focusing on two aspects of a research where the first is the level of consistency of a measure of a concept (internal) and if the result of the research is repeatable (external). (Bryman and Bell 2007). In this research the author is dealing with a topic which has received a lot of attention both from the academic world and the industry. As a result the measures and frameworks used has been aligned and standardized which makes them more generic than a set of specific indicators.

Another important issue when discussing reliability is concerned with the sources. During this research the author has done his outmost to access reliable sources and collect relevant data. In this case the author has been dealing with well known topics within the industry which has simplified the process of finding relevant information. The author has also been very selective in identifying reliable source which are well known within the academic world. The collection of data was based on previous research done in the field of IT-management, Enterprise Architecture, SOA, and EDA.

As concerns the reliability of this work, we believe that if any other research with the same topic, and with the use of same resources will come up with the same results.

2.2 Applied research strategy

Patel and Davidson (1994) enumerate three main steps when conducting a research. They start with the understanding and definition of the problem, continue with how it shall be studied

and what approaches will be used, and end with preparation and execution of the research. The following figure provides a visual explanation of the research approach applied in this study which is applying the approach recommended by Patel and Davidson (1994).

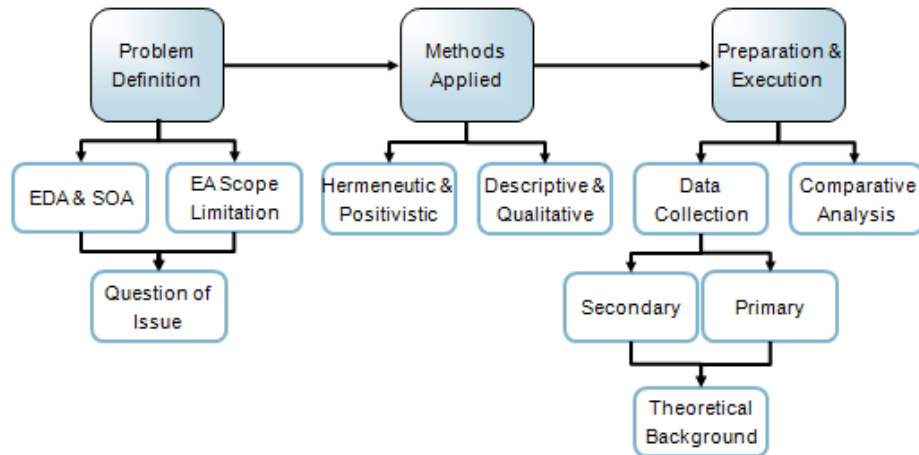


Figure 2 Research Process

The problem definition was initiated by the process of revising personal experience and reviewing literature from previous and ongoing research. The aim of the problem definition phase was to gain a deeper knowledge and understanding of the topic, identify reliable sources of research, scope specific areas of interest, and identify if similar questions have been raised and studied by others. This pre-study was of great help and value before determining the final research question.

Having the research question in mind, the research approaches discussed in previous sections were considered. Due to the fact that this is a study where existing material is used to describe the topic in detail and focusing on a limited number of aspects it was obvious to choose and apply a descriptive and qualitative approach. To be able to conduct a reliable and valid research the analysis had to be based upon some frameworks and theoretical background. Due to the nature and context of the topic it was decided to study and use IT-management, Architecture, and enterprise architecture as the theoretical base. Enterprise architecture frameworks were used to scope, analyze and compare SOA and EDA.

After analyzing and compiling secondary and primary data in the theoretical background, the characteristics of SOA and EDA were identified. Next step was to perform a comparative analysis. In the comparative analysis EDA and SOA characteristics are summarized but the discussion will go further including their relations as well.

The comparative analysis is comparing EDA and SOA from different points of view and aspects. EA frameworks from several sources were used to identify the different aspects. The first source used was Magoulas and Pessi (1998) where a similar comparative analysis between two architecture styles is conducted. The aspects used by Magoulas and Pessi (1998) are divided into two different categories.

1. Main characteristics – where the following views are covered: Business, Information, Information System, and Information System architecture.
2. Guiding principles – where some of the main guiding principles for each architecture style are identified and discussed.

The selection of aspects can always be questioned but my ambition is not to do a complete comparison including all areas. My intention is to focus on main characteristics and guiding principles, differences between the two architecture styles and how they interact and affect each other in a common environment.

3. Theoretical Background

In this chapter concepts relevant to the selected question of issue are presented. IT-management, Architecture, Enterprise Architecture, and Enterprise Architecture Framework are discussed and described. This chapter starts with discussing IT management which is the effort of planning and using information technology resources within an enterprise. The chapter continues with describing architecture, enterprise architecture and frameworks as a tool for applying and supporting IT management.

3.1 IS/IT Management

Many organizations are today dependent on their information systems which are crucial success factors for the business. This dependency is nowadays very obvious but has not been as obvious during the last decades. Since the start of use of information systems during the 1950s the scope of IS/IT management has been growing. During the early days managers have been mainly interested in Information Technology (IT) needed to provide some fundamental support as automation and faster data management. By information technology in this context we are refereeing to technical capabilities and functions they provide. (ward and Peppard 2002).

Since 1950s the use of IT is increasing in many areas e.g. facilitating fundamental changes in daily work, integrating functional activities on all levels and between organizations, increasing competitiveness and creating new strategic possibilities. The growing importance of IT has required increasing level of long-term planning. The first step towards long-term planning of IT was taken during 1980s. During this period the focus was also gradually turned into the use of information systems (IS) and the needs of users. It was evident for many managers that only focusing on IT will not lead to success. What distinguish organizations with high performance information systems is not only technical capabilities but the way they manage to plan, use, and deliver business support. Organizations should concentrate on business, while considering information systems and technology as part of the solution. Information systems should be treated and implemented efficiently for business to survive and provide competitive edge. During this period new ideas and ways of using information systems where established and implemented focusing on business e.g. business process redesign supported by information systems, improved business integration, better use of information while making business decisions, etc. (ward and Peppard 2002).

As the role of information systems has grown the need for strategic planning and alignment with business has evolved. During 1980s the bottom-up approach where IT was used without any strategic planning was extended by a top-down approach. The new approach required new way of thinking and planning when dealing with IS/IT management. The cornerstone in the top-down approach is the business strategy which is the main input when defining an IS/IT strategy. (ward and Peppard 2002).

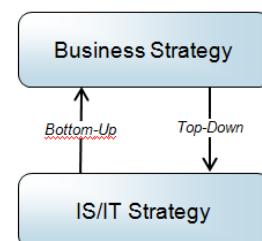


Figure 3 Strategic alignment

“The IS strategy defines the organization’s requirement or demand for information and systems to support the overall strategy of the business. It is firmly grounded in the business, taking into consideration both the competitive impact and alignment requirements”. (Ward and Peppard 2002:44)

In this definition alignment with business is a cornerstone and definite part of the IS/IT strategy definition. The need for alignment is also increasing as a business moves into more innovative and frequent use of information systems. Another important aspect of alignment is the bottom-up alignment where IS/IT strategy can be an enabler for business strategy and new business opportunities. A successful IS/IT strategy definition and alignment requires close and continuous collaboration between business and IT managers and the sooner this process is initiated the bigger are chances for success. Developing an IS/IT strategy includes identifying and planning long-term activities including both information systems and information technology to get implemented. Over the years different approaches have been used when developing IS/IT strategies. The approaches have been categorized by Earl (1989) into the following five groups:

1. Technology led – applies a bottom-up approach where IT specialist are focusing on IS/IT capabilities and how they are mapped into existing environment. This will lead to a higher degree of understanding of existing IS/IT environment but the strategically IS/IT advantages are limited.
2. Method driven – is starting by defining, analyzing, and prioritizing business needs which are translated into IS/IT initiatives and plans. This is a top-down approach.
3. Administrative – is a mix between top-down and bottom-up approach focusing at a detailed IS/IT plan which is accepted by both business and IT.
4. Business led – is initiated by senior management to define an IS/IT plan based on existing business strategy with the aim to achieve strategic and competitive advantage.
5. Organizational – this is perhaps the most mature approach to use when aligning business and IS/IT strategies. This approach is a mix of all previous ones aiming at IS/IT strategy definition and alignment together with business organization.

Analyzing each approach will help organizations to assess and increase the level of alignment between existing business strategy and IS/IT strategy, and also set the success level of IS/IT management. As the organization is trying to increase the level of alignment new approaches need to be implemented. However in order to achieve full and relevant alignment some earlier approaches need also to be maintained.

By summarizing this section we are able to come to some conclusions about the evolution of strategic IS/IT management. The first one is that the evolution of information systems and technology has increased its strategic focus in the enterprise. This leads us to the second conclusion which is emphasizing on the importance of IS/IT strategy and its alignment with business. The third and final conclusion is the need of new and more extended IS/IT management including strategic planning, structuring, and delivering. The main purpose of IS/IT management is to put in place a set of actions aiming at increasing the long-term health and optimal usage of existing information technology resources, attaining competitive strength. The aim and purpose of IS/IT management is also well summarized by the following definitions.

Strategic IS planning is the continuous review of computer technology, applications and management structure to ensure that the current and anticipated information and

process needs of the organisation are met in a way that provides an acceptable return on investment, is sensitive to the dynamic politics and culture of the organisation and is aware of the sociological environment within which the organisation exists. (McBride 1998:228).

“Information Technology Management is concerned with exploring and understanding Information Technology as a corporate resource that determines both the strategic and operational capabilities of the firm in designing and developing products and services for maximum customer satisfaction, corporate productivity, profitability and competitiveness.” (Wikipedia, the free encyclopedia, 12th November 2009)

The need to practice IS/IT-management is automatically leading us to search for a set of tools and methods facilitating IS/IT management. Architecture has for many years been used by managers and designers to model common patterns helping to understand existing structures and identify belonging components. The use of architecture when planning and designing Information Technology aligned with business is known as Enterprise Architecture. The field of enterprise architecture started in 1987 with the publication of an article in the IBM Systems Journal titled “A framework for information systems architecture”, by J.A. Zachman. Soon after Zachman released the first ever enterprise architecture framework. During late 1980s Zachman had come to the conclusion that effective IS/IT management requires structure, planning, and architecture. His answer to this problem was enterprise architecture. (Zachman 1996). In the following sections architecture and enterprise architecture will be discussed and how they can be used as supporting tool when managing information technology.

3.2 Architecture

The need for architecture and structure is today very obvious within the Information System and Information Technology area, but nothing new in other business. (Aerts et al. 2003). Before discussing Enterprise Architecture we would like to introduce some related definitions in this section.

The need for planning and structure is also obvious when constructing buildings or cities. Building a small cottage is perhaps not that complex while building New York City is much more complex requiring several architects and much more planning. The relationship between the complexity and planning for cities and building are much comparable with the complexity and need for structure when planning information systems. Building a large, complex information system without architectural vision is like trying to build a city without a city planner. Another important distinction between building a cottage compared to a city is that a city is a dynamic environment with continuous minor and major changes. While building a cottage is a short-lived project with defined resources and time limit.



Architecture is a tool box providing necessary tools and methodologies to develop the overall architectural vision for a city, organization, or information system. (Sessions. 2007).

Another very important benefit of architecture except structure is increased flexibility. Flexibility is a significant architectural quality since it supports the adaptation of business or information systems to different situations. Flexibility is achievable at two different levels. The top level is architectural flexibility where new changes are supported by modifying the architecture e.g. the relations between components or the organization. The bottom level is where component flexibility is achieved. At this level new changes are supported by updating existing components or adding new components within the existing architecture. (Aerts et al. 2003).

At this point it is much relevant to define some architectural terms which will be used later in this paper. Every time these terms are used further on in this study the reader can refer to the explanations in this section. The definitions are taken from Session (2007).

“Architect – one whose responsibility is the design of an architecture and the creation of an architectural description.” (Session 2007:5).

“Architecture – the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.” (Session 2007:5).

“Architectural artifact – a specific document, report, analysis, model, or other tangible that contributes to an architectural description.” (Session 2007:5).

“Architecture framework – a skeletal structure that defines suggested architectural artifacts, describes how those artifacts are related to each other, and provides generic definitions for what those artifacts might look like.” (Session 2007:5).

3.3 Enterprise Architecture

As architecture has been used for many centuries to plan houses and cities Enterprise Architecture is the art of planning and structuring information technology aligned with business needs and business strategy. Enterprise architecture is today used as a mean supporting IS/IT management efforts. Zachman is of the opinion that the issues of quality and change are the circumstances that are forcing us to emphasize the issues of structure or Enterprise Architecture. This will not happen by accident or through one or several technology implementations. It will only happen because of a reasonable and long-term investment in developing and maintaining enterprise architecture. (Zachman 1996).

Whittle and Myrick (2005) are emphasize that many enterprises lack a formal business structure and present a point of view that enterprise is not about chaos. It is about connectivity and understanding relationships to both internal and external factors. The need and importance of enterprise architecture to achieve business goals is obvious also from their point of view. Enterprise architecture will provide necessary support to allow a central understanding and create link between the strategy, its supporting architectures, and its planned activities. The following figure is illustrating main links uniting an enterprise. Enterprise architecture is a key component providing support to reach a unified and integrated enterprise.

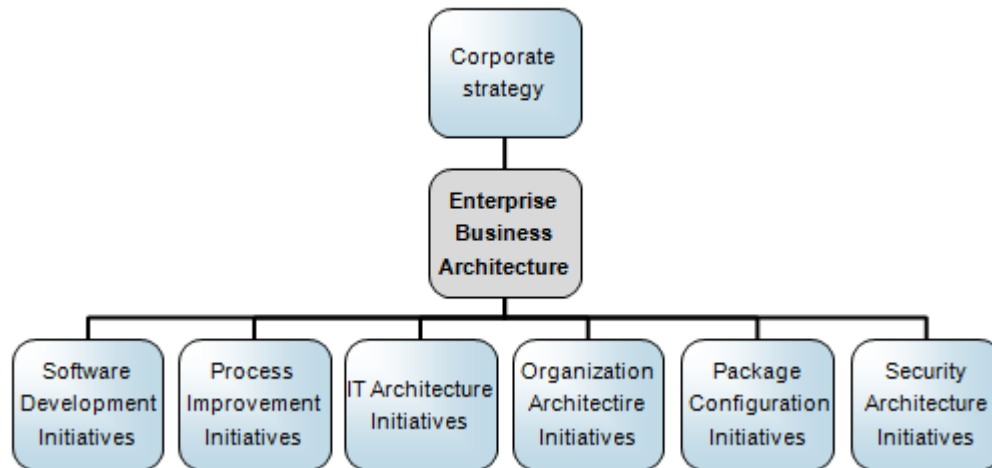


Figure 4 Formal Links (Whittle and Myrick 2005)

Though business architecture has a major role in the model above it is not the only artifact in an enterprise architecture framework. Over time the scope of enterprise architecture has changed and expanded to include business, information, application and technology domains including hardware. The increasing scope and complexity is enforcing more structuring and break-down in several domains. These domains will be described later in this thesis. (Aerts et al. 2003). The two following descriptions of enterprise architecture have been selected to be presented in this thesis concluding this section.

“Enterprise Architecture – is that set of design artifacts, or descriptive representations, that are relevant for describing an object such that it can be produced to requirements (quality) as well as maintained over the period of its useful life (change).” (Zachman 1996:5).

“Enterprise Architecture – a strategic information asset base, which defines the mission, the information necessary to perform the mission and the technologies necessary to perform the mission, and the transitional processes for implementing new technologies in response to the changing mission needs. Enterprise architecture includes baseline architecture, target architecture, and a sequencing plan.” (CIO Council 2001:5).

3.3.1 Enterprise Architecture Frameworks

As a pioneer and creator of the most famous and first enterprise architecture framework Zachman has had a major impact on designing and deciding the content of Enterprise Architecture. Zachman did study different real-life product development cases and concluded that the representations of the interesting characteristics are of *what* the product was made of, *how* the product functions, and *where* the components are located relative to one another. It was obvious that a complete description of the case should also include descriptions of *who* performs what relative to the product, *when* things happen and *why* various product choices being made.

Dealing with all the characteristics at one time would result in a very complex and useless picture. It is a process of “abstraction”, a simpler version on which to focus for some particular exercise. What Zachman did was to take his generic framework that he developed through observation of the descriptive representation of physical products and applied it to an Enterprise to produce the framework for enterprise architecture. The result was a framework

with a generic classification scheme for descriptive representation of any object. (Zachman 1996).

	What	How	Where	Who	When	Why
Scope						
Owner						
Designer						
Builder						
Sub-contractor						
Product						

Figure 5 Zachman Enterprise Architecture Framework¹ (Zachman 1996:7).

The framework is of great benefit during documentation and communication. The framework is standardized and will simplify communication through a standard set of vocabulary and notations which are commonly understandable by all users. Another reason for using enterprise architecture is aligning business requirements with IT initiatives. Strategic alignment has been studied and discussed seriously since 1980s. Strategic alignment will not be covered further in this chapter but it shall be mentioned since it has played a major role in the definition and designing of enterprise architecture frameworks. Henderson and Venkatraman (1989) developed Strategic Alignment Model (SAM) which later was enhanced by Maes et al. (2000) producing another enterprise architecture framework called the Unified Architecture Framework (UAF). UAF incorporates additional functional and strategic layers into the model to reflect the current need for information and communication. The unified framework is a generic framework for investigating and relating different components of information management, and deals with the links of business, information, communication and technology at the strategic, structural and operations levels. This framework is the first real attempt to refine SAM to reflect the fact that IT and business strategies are moving closer together as technology evolves and becomes more integrated. (Avison 2004).

There are clear similarities between Zachman's framework and UAF. Both frameworks are considering different characteristics and abstraction levels.

¹ For a more detailed description of the framework see Appendix A

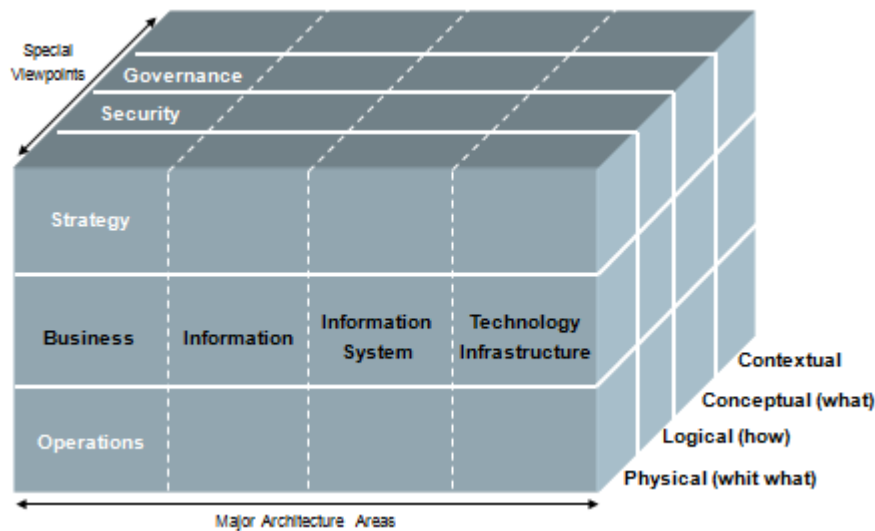


Figure 6 Unified Architecture Framework (Maes et al. 2000:19)

Other enterprise architecture frameworks worth mentioning are The Open Group Architectural Framework (TOGAF) (The Open Group, 2003), The Federal Enterprise Architecture Framework, (FEAF) (CIO-council, 1999), and Treasury Enterprise Architecture Framework (TEAF) (Department of the Treasury CIO Council, 2000). The frameworks discussed so far are also revealing the content of enterprise architecture. Looking at UAF we can identify four major architecture domains which also can be recognized from Zachman's generic framework. The four domains are business, information, information systems, and technology infrastructure.

3.3.2 Business architecture

Business architecture is describing business objects, functions, processes, actors and how the business shall function. Business architecture is about defining and designing your business to be able to take full advantage of the resources and capabilities you possess. The business architecture is derived from the business vision, goals and strategies. Structural or organizational architecture is also mentioned in combination with business architecture. Nadler and Gerstein (1992) define organizational architecture as the art of shaping behavioral space to meet the needs and aspirations of a business. (Aerts et al. 2003). Enterprise business architecture defines the enterprise value streams and their relationships to all external entities and other enterprise value streams and the events that trigger instantiation. It is a definition of what the enterprise must produce to satisfy its customers, compete in a market, deal with its suppliers, sustain operations, and care for its employees. It is composed of models of architectures, workflows, and events. (Whittle and Myrick 2005).

In an interview done by Howard (2000) with former CEO of Xerox Paul Allaire it is obvious that business architecture is a necessity to be able to cope with all new market challenges. Many companies are aware of the need but few have approached the process of business architecture or organizational redesign systematically and methodically. Over the years Allaire has created a new business giving the enormous flexibility and ability to adapt to new requirements very faster than the competitors.

"A big advantage of this structure is that it is remarkably easy to adapt. When we see new markets emerge or new technologies that don't fit into our current structure, we can simply add another business team or even a whole new division. Similarly, we can split

a business division – or even eliminate one altogether – without changing the basic architecture of the company”. (Howard 2000:112)

According to Aerts (2003) business architecture defines the business system in its environment of suppliers and customers. The system consists of humans and resources (including IT), business processes, and rules. It belongs to the disciplines of industrial engineering and management science.

3.3.3 Information architecture

Kettinger (1992) refers to the early days of system development where data was not formally defined outside the program. During the years the concept of data independence brought the realization that the center of data processing was the data rather than the application. Soon the need for structured and common data structures was obvious and the basic blueprints for an organization’s information architecture were created. Kettinger (1992) defines information architecture as

“A high level model of a set of databases configured to support the organization’s value-adding business processes. The model may be portrayed in graphical, tabular, or narrative form and is independent of technology and current organizational structure.” (Kettinger 1992:83).

Bracheau and Wetherbe (1986) have similar description of information architecture.

“Information architecture is a high level map of the information requirements of an organization. It is a personnel-, organization-, and technology-independent profile of the major information categories used within an enterprise.” (Bracheau and Wetherbe 1986:453f).

Information architecture helps you redesigning your business and processes. It will provide many with a lot of valuable data e.g. information objects, users of data, security issues regarding data handling, availability, and storage needs.

3.3.4 Information System architecture

Information Systems (IS) architecture is perhaps the most common architecture area in organizations. Also called application architecture it details the software application components and their interaction. (Aerts et al. 2003)

Magoulas and Pessi (1998) describe two theories of IS-architecture design that have dominated the professional and scientific debates in Sweden: the enterprise based design theory and the information based design theory. The enterprise based design theory stresses coordination between information systems, where each information system is administered by the part of the enterprise that uses that system. The information based theory of IS architecture design has as its starting point the premise that information is a central resource that must be controlled centrally. While there are similarities between the two theories of design, there are also significant differences. The main similarity is an ambition to improve the overall supply of information in the enterprise. The most fundamental difference is with regard to information, accountability and principles for integrating information systems.

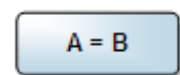
According to Aerts (2003) IS architecture has become highly distributed, consisting of relating components that hold functionalities. The mapping of these components is flexible

and described in an implementation. Mapping and integrating components or systems is a main part of the IS architecture often referred to as Integration Architecture. The business model has stressed the importance of integration and during the years integration of information from different applications has gradually changed from manual to automatic. The approach widened from expanding and integrating applications to the integration of applications based on different architectures. The scope of integration has also widened from integrating applications based on a single, homogenous information model to the integration of applications based on different, heterogeneous information models.

Hohpe and Woolf (2006) leave the definition of integration very broad and describe it as connecting computer systems, companies, or people. They refer to application integration as the task of making disparate applications work together to produce a unified set of functionality. These applications can be custom developed or purchased from third-party vendors. They likely run on multiple computers, which may represent multiple platforms, and may be geographically dispersed. Some of the applications may be run outside of the enterprise by business partners or customers. These issues and others like these make application integration complicated and increase the need for structured integration and integration architecture. Nowadays the issue is not the technical infrastructure or platforms. The main obstacle when integrating information systems is different data semantics, data quality, data synchronization, etc.

Magoulas and Pessi (1998) are also discussing integration architecture and point to the growing numbers of systems within enterprises and the following challenges. Number of systems within enterprises is growing through mergers and acquisitions which need to be maintained, updated or removed. Existing systems need to function and interact with new systems and businesses. Much of the legacy has been introduced for different purposes. In some cases systems are working isolated from other systems while in some cases they are very tightly connected and require high level of integration. All these are integration challenges that need to be considered. According to Magoulas and Pessi (1998) integration is the process aiming at creating appropriate coupling between different environments, systems, components, etc. to be able to increase the cohesion, enable coordination and streamlining collaboration between systems. They refer to four different types of integration.

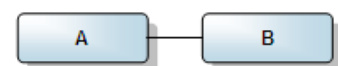
Unified systems are based on same standards. It may be on-shelf standard products. If one unified system is modified it will also affect the other systems in the same package.



Coordinated systems have one or several shared components. If a shared component in one system is modified it will affect all systems using that component.



Coupled systems have some kind of interaction in between. This can e.g. take place by using messaging. The systems are not sharing any components and are not affected by changes in the opposite system unless the communication is modified.



Independent systems have no interaction or shared components, which means that changes in one system will not affect any other surrounding system.



What makes good information system integration? If integration needs were always the same there would be only one integration style. However like any other complex technological effort application integration involves a range of considerations and consequences that should be taken into account for any integration opportunity. (Hohpe and Woolf 2006).

Due to the importance of integration when discussing architecture we will add it and discuss it separately within the analysis chapter.

3.3.5 Technology Infrastructure architecture

Though this architecture layer is not within the scope of this study a short description is included to cover the entire enterprise architecture framework. Technology architecture is the architecture of the layer, which describes the computers, networks, peripherals, operating systems, data base management systems, hardware, middleware, etc. that will be used as a platform for the construction of the information systems for the enterprise. Its description includes various platform paradigms such as mainframe– terminal, n-tier, client–server, etc. (Aerts et al. 2003)

3.4 Two architecture concepts

So far in this thesis we have discussed the need and purpose of IT-management and architecture. Two architecture concepts are SOA and EDA. Both architecture concepts will be described within the context of the four enterprise architecture domains.

SOA is an architecture style, an idea or approach, of how information technology can be planned, designed, and delivered as modular business services to achieve specific business benefits. SOA is providing specific guidelines and principles to structure both business and information technology architecture. (Marks and Bell, 2006). Business services are the corner stone within service oriented architecture.

“In the context of enterprise architecture, service-orientation, and service-oriented architecture, the term service refers to a set of related software functionality, together with the policies that should control its usage.” (Wikipedia, the free encyclopedia, 3rd March 2010)

Event Driven Architecture is another architecture style with its own design principles. EDA is characterized by the development of a set of relatively independent actors who communicate events amongst themselves in order to achieve a coordinated goal. While SOA is focusing on structuring business services, EDA is focusing on structuring business events. This is done within all four domains included in an enterprise architecture framework i.e. business, information, information system, and technology infrastructure. In order to understand event-driven architecture we must understand the definition of an event which is the DNA of EDA.

“Event is a notable thing that happens inside or outside your business. An event (business or system) may signify a problem or approaching problem, an opportunity, a threshold, or a deviation.” (Michelson 2006)

Both SOA and EDA differ from previous architecture and design concepts by breaking business into smaller and reusable components. These components are translated and implemented into IT components that are loosely coupled and can be integrated in a dynamic

and flexible manner. SOA and EDA try to replace and break old and large legacy systems and architectures into more flexible and reusable business and IT components.

3.5 Summary

Today, organizations and enterprises must adapt and adapt quickly. The fast moving and agile business environment put enormous pressure on the flexibility of strategies and operations. Increasing market requirements together with an increasing use of information systems are adding to the complexity of managing both business and IS/IT. That is why we need well defined business and IS/IT strategies providing a holistic view of objectives, plans, and priorities within an enterprise. However reaching a true holistic view is not possible without aligning business and IS/IT strategies. Getting IS/IT strategy aligned with business strategy and objectives is vital for the success of IS/IT performance.

As the importance and usage of information systems has grown during the last five decades the need for IS/IT management, structure and long-term planning has escalated. The answer to this long-term planning and structure is Enterprise Architecture. For most organizations enterprise architecture is the missing link between strategy and result. Enterprise architecture is the long-term, strategic approach putting this link in place.

Existing enterprise architecture frameworks can be used to provide a common language when defining architectures for business, application, information, and technology. The common language and notations are understood by all employees who are facilitating future changes and communication. Enterprise architecture makes the components at different layers to fit and integrate into a holistic view.

4 Service-Oriented Architecture (SOA)

In this chapter main characteristics of Service-Oriented Architecture covering both the objectives of this concept and its building blocks are explained and discussed.

4.1 Background

SOA is a very relevant concept within this period of time. It is a concept with great promise for IT, business, and for organization as a whole. SOA cannot be summarized as a product, or a solution, or a technology. SOA cannot be reduced to a couple of software products. SOA is not a quick fix for the IT complexity. SOA does not address every IT challenge facing business and IT executives today. And last but not least, SOA is not dead as some are claiming that! So, what is SOA? There are many examples of SOA definitions. Most of these definitions focus on the technical aspects of architecture, although some include business characteristics. These definitions listed here from multiple sources are interesting because they illustrate the variety of different views or expressions of what an SOA is.

A business definition: “SOA is a conceptual business architecture where business functionality, or application logic, is made available to SOA users, or consumers, as shared, reusable services on an IT network. “Services” in an SOA are modules of business or application functionality with exposed interfaces, and are invoked by messages.” (Marks and Bell 2006:1)

Another business definition (IBM): “A Service-Oriented Architecture is an enterprise-scale IT architecture for linking resources on demand. These resources are represented as business-aligned services which can participate and be composed in a value-net, enterprise, or line of business to fulfill business needs. The primary structuring element for SOA applications is a service as opposed to subsystems, systems, or components.” (Mamdouh 2001)

A technical definition (Gartner): “Service-oriented architecture is a client/server software design approach in which an application consists of software services and software service consumers (also known as clients or service requesters). SOA differs from the more general client/server model in its definitive emphasis on loose coupling between software components, and in its use of separately standing interfaces.” (Mamdouh 2001)

Another technical definition (W3C): “A set of components which can be invoked, and whose interface descriptions can be published and discovered.” (Mamdouh 2001)

An integration focused definition: “A Service-Oriented architecture is a framework that supports the discovery, message exchange, and integration between loosely coupled services using industry standards. Each party complies with agreed on protocols and carries out its part in the overall execution of processes involving services from diverse organizations.” (Khoshafian 2007)

The intention with the above definitions is not to describe what SOA is but to demonstrate different aspects of SOA. None of the descriptions above are wrong, nor are they right one by one!

4.2 Characteristics of SOA

Businesses today cannot survive and prosper without using different technologies into both their day-to-day operations and their long-term strategy. Enterprises are forced into broader connectivity and increased revenues, but they also focus on innovation by restructuring applications for greater flexibility and lower costs. Given the rate at which the market is changing, it is almost impossible to carry out long-term planning. It is critical to be able to plan rapidly and continuously. One of the main goals of SOA is to provide a response to that agile enterprise. (Bieberstein et al. 2006)

According to Marks and Bell (2006) SOA is a business concept, an idea or approach, of how IT functionality can be planned, designed, and delivered as modular business services to achieve specific business benefits. SOA is a concept that has direct business advantage for all organizations. For the business, SOA means increased customer satisfaction, real business agility, faster time to market, ease of partnering, and lower business cost. For IT organizations, SOA means faster time to market, greater asset reuse, greater productivity, lower IT cost, and agility. SOA benefits an IT organization through faster application development, lower overall costs, greater soft asset and services reuse, higher-quality applications, and overall faster response to business customer requests for system enhancements and modifications.

Service orientation also provides the ability to loosely couple applications, trading partners, and organizations. In addition to that independent services can be composed in processes to provide even greater value. SOA promises to finally realize “agility” for organizations. The word agility is often mentioned when discussing SOA and can be described in two forms. The first one is the ability to change business processes to meet changing market demands and customer requirements, and reduce costs; and the second one is the ability to execute business processes faster or launch new processes, products, and services faster. Agility and speed are both real and tangible benefits of migrating to SOA and reusable services.

ROI and value of SOA has been abstract and hard to prove for many years. However during the last couple of years due to increasing maturity of SOA more and more focus has been directed towards measuring the value of SOA and value metrics definition. In a survey conducted by Gartner on the topic “The value of SOA” it was proven that SOA is creating value for organizations that follow it. This value is mostly in improvements to agility and rapid ROI. More than 60% of organizations said that their SOA projects had a positive impact on the organizations' ability to grow revenue. Other key findings were that SOA projects generate positive returns typically within 10 months, SOA reduces the cost of building IT information systems, and nearly 50% of the respondents said that SOA has helped to increase revenue for business. (Schulte and Sholler 2009)

4.3 Concept

SOA is an architectural style whose goal is to achieve loose coupling among a set of services in relation. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by organizational units as well as software agents on behalf of their owners. An SOA is a classic Request/Reply type architecture implemented in a one-to-one relationship. A service consumer invokes a service provider through the network and has to wait until the completion of the operation on the provider side. (Marechaux 2006)



Figure 7 Request/reply mechanism in a SOA

Service orientation consists of a number of business services that support a flexible and dynamically configurable business processes. The famous triangle is often used to illustrate the concept of Service Orientation. There are three essential elements for service interaction: (1) registering the service to a registry; (2) locating the service; and (3) making service calls and message exchanges. A service is offered by a provider to a consumer through its interface. An interface describes the contract between the provider and the consumer. In other words, it should specify what the provider is obliged to do on behalf of the consumer and what responsibilities the consumer agrees to in using the interface. (Allen 2006) A service registry is a network-based directory that contains available services. It is an entity that accepts and stores contracts from service providers and provides those contracts to interested service consumers. (McGovern et al 2003)

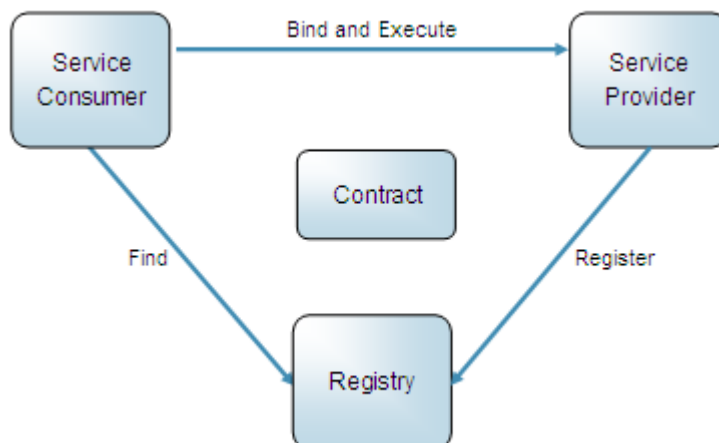


Figure 8 Concept of Service Orientation

Further have Marks and Bell (2006) described the elements within SOA which they present in a model. All elements are important and must be realized to achieve a successful SOA implementation. Each essential ingredient is explained in this section.

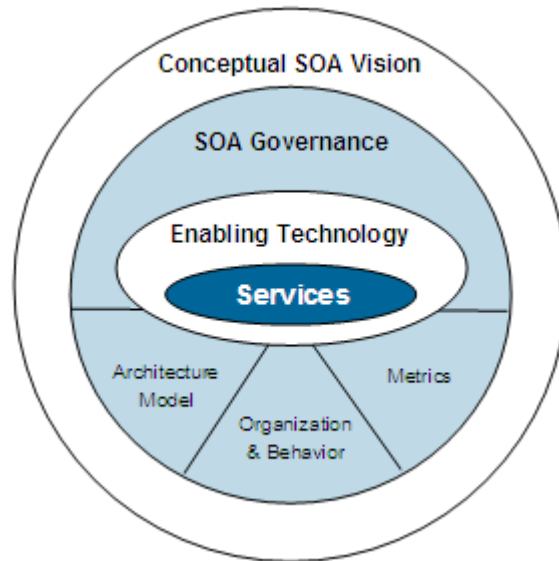


Figure 9 Elements of SOA (Marks and Bell 2006:6)

Conceptual SOA vision – *“includes clearly defined business, IT, and architectural goals, and a governance model and policies to help enforce standards and technical requirements of the SOA over time. This is the definition of an SOA target state, the goal to be achieved over time.”* (Marks and Bell 2006:2)

Services – In order to understand service-oriented architecture you must understand the definition of a service which is the DNA of SOA. Services are the primary architectural asset of an SOA and can cover any possible service in an organization. The fundamental unit of an SOA is a service. Services are reusable modular units of business capabilities, processes, or technical functions that are accessed and delivered in a repeatable fashion to consumers of those services. Services in order to meet the needs of the organization, must meet certain criteria to provide the most value to the organization. (Marks and Bell 2006)

- Services should represent business functions, processes, or transactions and include other components or services within them.
- The contracts between services must be well-defined and separate the functionality from its technical implementation. Along with services comes a service design model to assure reusability, interoperability, and integration across all business processes and technology platforms.
- The services must be “loosely coupled” which means designing services such that specific implementations of services can be replaced, modified, and evolved over time without disrupting the current service consumers and the overall activities.
- Services should be discoverable. This means not only that the services are designed well, but that their contracts are published and visible to an intended audience.
- Services should be durable and map to lasting business or process.
- Services should be designed in a composable way to be able to be incorporated into other services as necessary.
- Services should be business aligned. Service analysis and identification should begin with business imperatives and business requirements.
- Services must be reusable across and within business processes and have multiple consumption patterns.

Allen (2006) describes a service as a functionality that must be specified in the business context. It is of great importance to focus on the how relevant the service is for the consumer and what functionality it is delivering. Implementation details should not be in focus and revealed. It is neither a necessity to automate the implementation of a service – it could consist of purely human activity.

Enabling technology – *“is essential to support realization of your SOA vision. However the enabling technology is not your SOA. The enabling technology must be implemented to accomplish two objectives: (1) it must allow your services to operate reliably and securely in your enterprise in support of your stated business objectives; and (2) it must enable you to carry forward your existing IT architecture as well as enable legacy systems to be leveraged to support your SOA goals. In many organizations, legacy systems are major contributors of services to an SOA.”* (Marks and Bell 2006:3)

SOA governance and policies – *“define the various governance processes, organizational roles and responsibilities, standards and policies that must be adhered to in your SOA conceptual architecture. An SOA conceptual architecture cannot be realized unless it is communicated to business user, developers, architects, business and IT executives, business analysts, and close trading partners.”* (Marks and Bell 2006:3)

Metrics – *“are needed to measure the results you are achieving. These metrics include service-level agreements (SLAs) for individual services, as well as usage metrics, developer metrics, business and return on investment (ROI) metrics, and process metrics. The metrics shall be planned early and be used as soon as going live.”* (Marks and Bell 2006:4)

Organizational and behavioral model – *“is about the organizational behaviors, business decisions, and architectural choices which have affected the IT architecture. In order to achieve SOA behavioral and organizational considerations must be understood and changed first. The next step is to over time migrate toward SOA. New organizational models and behavioral models will be essential for SOA success.”* (Marks and Bell 2006:4)

All SOA elements described above are important and of major importance to have a successful SOA implementation in place. Unfortunately in many cases one or several of the elements are missing and most of the focus is on technical implementation aspects.

4.4 Principles

SOA has as any other architecture style a number of well defined principles which has to be implemented and followed. Here is a list of the standard principles when discussing SOA.

Loose coupling - one of the biggest benefits of services is their potential for loose coupling. Coupling refers to a connection or relationship between two things. This principle advocates and emphasizes on reducing (“loosening”) dependencies between the service contract, its implementation, and its service consumers. (Erl 2007) This principle is also discussed by McGovern (2003) who groups relations into two groups: loose and tight. Loosely coupled modules have a few well-known dependencies. Tightly coupled modules have many unknown dependencies.

SOA accomplishes loose coupling through the use of contracts and bindings. A consumer asks a third-party registry for information about the type of service it wishes to use. The registry

returns all the services it has available that match the consumer's criteria. The consumer chooses which service to use, and calls the method on it, based on the description of the service provided by the registry. The consumer does not depend directly on the service's implementation but only on the contract the service supports.

Although coupling between service consumers and service producers is loose, the implementation of the service can be tightly and in contradiction with the goal of loose coupling and code reusability. For instance, if a set of services shares a framework, a database, or otherwise has information about each other's implementation, they may be tightly coupled. (McGovern et al 2003)

Standardization - SOA standards are open in the sense that any software manufacturer has the right to use those standards when developing a SOA architecture. In addition, the process of creating and revising the standards is more or less democratic where any interested party has the right to participate in all meetings that lead to decisions about a standard. (Margolis and Sharpe 2007)

Standardized service contracts are also part of this principle. This principle is about making sure that all services within the same service inventory are in compliance with the same contract design standards. (Erl 2007)

Modularity – SOA implements services supporting well defined and modular business functions and information. These modules can later on be reused and be part of an end-to-end business process. The modularity of a service refers to the breaking of an application into many smaller modules. Each module is responsible for a single, distinct function within an application. (McGovern et al 2003)

Composability - the ability to effectively compose services is a critical requirement for achieving some of the most fundamental goals of service-oriented computing. Services are expected to be capable of participating as effective modules. (Erl 2007)

The modular composability of a service refers to the production of software services that may be freely combined with other services to produce new information systems. Modular structure enables services to be assembled into information systems the developer had no notion of when designing the service.

A service may be composed in three ways: information system composition, service federations, and service orchestration. An *information system* is typically an assembly of services, components, and application logic that binds these functions together for a specific purpose. *Service federations* are collections of services managed together in a larger service domain. For example, a checking account service, savings account service, and customer service may be composed into a larger banking-account service. *Service orchestration* is the arrangement and execution of one or more services in an organization. It is sometimes called a business process. It consists of multiple steps, each of which is a service invocation. If any of the service invocations fails, the entire transaction should be rolled back to the state that existed before execution of the transaction. (McGovern et al 2003)

Reusability - reuse is strongly emphasized within service-orientation; so much so, that it becomes a core part of typical service analysis and design processes. This principle is emphasizing service design and functional scoping that makes the service useable in more

than one scenario. The same service will be used within different business processes or information systems. (Erl 2007)

Discoverability - for services to be positioned as IT assets with repeatable ROI they need to be easily identified and understood when opportunities for reuse present themselves. The service design therefore needs to take the “communications quality” of the service and its individual capabilities into account, regardless of whether a service registry is in place or not. (Erl 2007)

Abstraction - this principle emphasizes the need to hide as much of the underlying details of a service as possible. Doing so directly enables the previously described loosely coupled relationship. (Erl 2007)

Coarse-Grained interfaces – the level of service granularity is a main success factor in Service-Oriented Architecture. However it is a very subjective topic to discuss since the levels of granularity are relative to each other. For instance, if a service implements all the functions of a banking system, then we consider it coarse-grained. If it supports just credit-card validation, we consider it fine-grained. In addition, if a method for inquiring about a customer returns all customer information, including address, this method would be coarser-grained than a method that does not return the customer's address.

The appropriate level of granularity for a service and its methods is relatively coarse. A service generally supports a single distinct business concept or process. It contains software that implements the business concept so that it can be reused in multiple large, distributed systems.

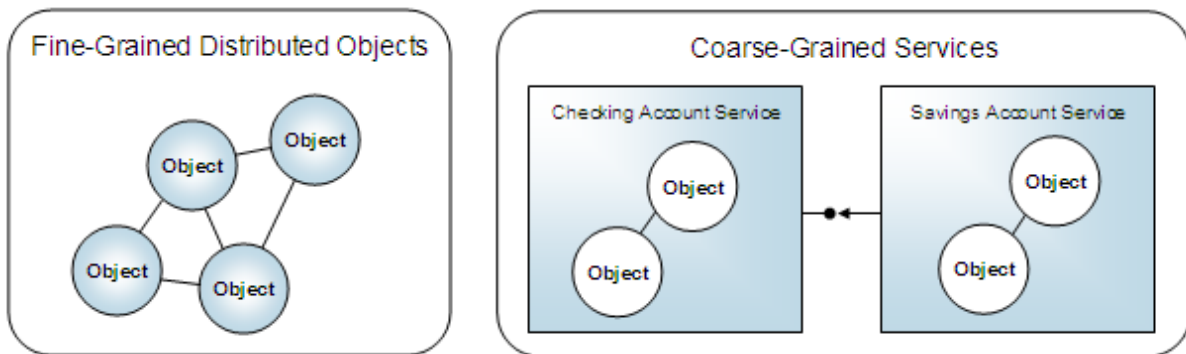


Figure 10 Coarse-Grained Services

The granularity of the service is a crucial design decision. If it is incorrectly predicted, consumers will have access to more functionality than they need. This can be a problem for security at the service level. It might not be possible to restrict a consumer from some methods and not others, only to the entire service. If this is the case, the entire service might have to be opened up to consumers. Granularity is a difficult problem to solve when designing service interfaces. It is important to understand the options and implement the most appropriate interface. Most often the focus is on determining the right granularity level of the service producer but attention should also be paid finding the right granularity level for service consumers. (McGovern et al 2003)

Interoperability - the primary benefit of SOA standards is that they make services interoperable, which means that services can communicate with one another, even if each

implementation is written in a different computer language or is accessed by way of a different transport protocol. (Margolis and Sharpe 2007) According to Erl 2007 interoperability is fundamental to every one of the principles we just described.

Statelessness - services are ideally designed to remain stateful only when required. Services should not rely upon long-lived relationships between consumer and provider, nor should an operation call rely on a previous invocation. (Erl 2007)

4.5 Data

Service oriented architecture represents a new way of thinking about everything in a company's IT structure, including how one thinks about data. It begins with the goal of achieving consistency between data sources. In order to achieve data consistency, you begin by separating your data from its tight dependency on the information systems that created it and update it.

Data is one of the organization's most valuable assets, but these critical data stores are typically separated into different data silos. Traditionally, business data has been managed in a way that tightly associates specific data definitions to specific information systems, such as finance, human resources, or ERP. The problem is that an organization's data resources were not designed for global use by all information systems. They were designed to suit one specific information system or, at best, two or three. When separate information systems gather their own data, simple errors in entering data make it difficult, and sometimes impossible, to aggregate the data that has been collected about a customer (or any other entity).

The silo approach when working with data may provide relevant information to a particular business unit, but it creates inconsistencies in data at an enterprise level. This happens because data is often defined to fit the precise view of a single business unit. How can you trust the information your business uses to make strategic decisions if poor data availability and quality keeps you from having a complete view of your customers or products? (Hurwitz et al. 2007)

In order to find a solution to all these data challenges SOA as an architecture concept includes some guidelines and patterns for how data in an enterprise should be managed. Three of the most important topics when discussing information management within SOA are perhaps data integration, data semantics, and meta data.

Data integration - In order to make data more reliable, consistent, and trusted, enterprises link data sources between departments or regions of their organization by using various data integration processes. Service oriented architecture is changing both the philosophy and the architectural framework for deploying the data integration software tools that manage the integration process. Implementing a SOA approach enables the business to access, manipulate, and share data across the organization in a consistent and technology independent way. This approach provides the business with more useful information to help make sound business decisions.

When organizations begin to apply SOA principles to managing their data assets, they move from fixing problems on the fly to delivering information as a service. Information as a service is an architectural approach that loosens the tight connections between data and information systems so that data can be controlled and shared across the enterprise. This

approach allows businesses to reach a consistent view of enterprise-wide information that has previously been very hard to achieve.

To provide information as a service to everyone in the business all the data the business people need must be treated consistently across the enterprise. Consistent definitions and rules for data must be based on the way the business as a whole needs to understand sales, customers, products, and profit. These rules are available in all organization and often called “business rules”. Information delivered as a service has been effectively certified by the enterprise as trusted data. This means you can trust that you and your counterparts across the enterprise are basing decisions on data that is secure, clean, and structured correctly. Everyone in the business is working with consistent rules about how the data is structured, accessed, and used. This common understanding of the data must extend across business units and regions to include information provided to partners and customers. (Hurwitz et al. 2007)

Data semantics - Other inconsistencies in data are based on semantic differences. Data semantics is the meaning of data. Semantics of data is used to ensure that everyone in the business has a common understanding of the business information and rules. Semantic differences in the use of basic business terms like customer, partner, department, is often a challenge for many organizations. Semantic interoperability is an architectural quality that measures how people and technology understand data and how this level of understanding impacts the exchange of information.

One of the main objectives of SOA is to make sense out of business chaos. This includes providing accurate information about the business to everyone involved in the business. One major step for making this happen is to ensure that each component of data can be used independently from its current technology and implementation. With service oriented architecture, you need to begin to think of data as a reusable resource which is called “information as a service”. (Hurwitz et al. 2007)

Meta data - The definitions, relations, and other characteristics used to describe how to identify, access, and use the company’s data are called metadata. Business services need to be able to access metadata in order to consume and deliver the data they need. Metadata is stored in the metadata repository containing definitions of business data and rules for mapping data to their actual physical location in the information system. This repository is in a technical layer between the actual data stores and the business services. The metadata repository is often referred to as a metadata layer because of its position in the information infrastructure.

The purpose of the metadata repository is to ensure that the data is of the right structure and quality before it’s consumed by a business service. The metadata repository also ensures that data from different sources can be linked together correctly. (Hurwitz et al. 2007)

4.6 SOA implementation components

To address the need for an enterprise-wide, service-oriented IT, companies and organizations have defined different reference models. One of these reference models is on demand operating environment (ODOE) defined by IBM (see Appendix B). In this thesis the ODOE reference architecture has been selected to help us identifying different components which are necessary in a complete service-oriented architecture.

Business Services - are the exposed part of the business processes and the business functionality that is accessed by and provides predefined value to the requestor. As previously indicated, the requestor can access these business services through a contract and interface. The requestor, in this case, can be a user within the organization, a user outside the organization's IT infrastructure or outside the organization altogether, or even a customer or partner completely separate from the organization.

Application Services - The Application Services domain comprises all of the components that are necessary to build a composite information system. This domain includes components to support user access, business function, common services, user interaction, business process choreography, and information management.

Common Services - The Common Services components enable personalization of the delivery and individual processing, as well as other utilities such as reporting. These services enable you to define and execute conditional flows, mainly across logical business function services but also to any other service. IBM recommended software model for services choreography uses the industry standard Business Process Execution Language (BPEL) specification.

Information Management Services - The Information Management Services provide a uniform way of representing, accessing, maintaining, managing, analyzing, and integrating data and content across heterogeneous information sources. They are an essential component of SOA and play a critical role in enabling SOA. There are many functions within information management, such as federation, replication, modeling, search, and analytics. Each of these functions can be provided as reusable and componentized services.

Enterprise Service Bus - The Enterprise Service Bus is an intermediation layer that interconnects all of the services, enabling all of the (loose) coupling characteristics.

Infrastructure Services - Finally the Infrastructure Services domain is a set of platform-independent services that enables all of the other services domain components to be installed, executed, and controlled on a concrete infrastructure combination of operating systems and network and hardware systems. (Bieberstein et al 2006)

4.7 Enterprise Service Bus (ESB)

To realize the scenario of an automated, self-managed SOA, ESB is an essential architectural component. It is also a core part and a key architectural element within ODOE or any other SOA reference architecture. The ESB, acting as an intermediary, supplies loosely coupled connectivity between the participants in service interactions and, provides the backbone of an SOA. The following figure shows the logical relationship between service requesters, service providers, and the ESB. Service requesters and providers interact by exchanging messages. The ESB is a logical component supporting interactions and providing loosely coupled interconnectivity between the provider and the consumer of a function. Its role as a logical intermediary allows the ESB to process messages as they flow between a service requester and service provider. This processing is called mediation. (Flurry 2007)

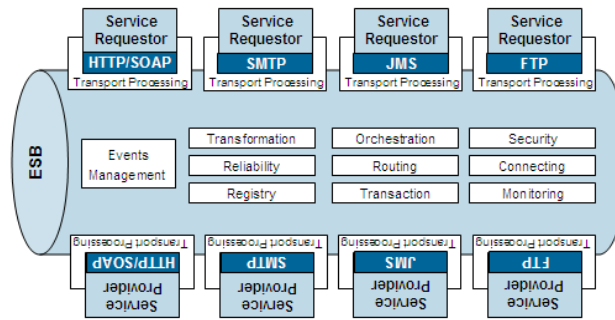


Figure 11 ESB architecture (Khoshafian 2007)

An ESB is a core intermediary that ties services together. ESB can be implemented in various ways, such as with classical messaging, EAI, and brokering technologies or by using platform-specific components e.g. integration buses in J2EE application server. The ESB can also be a combination of both EAI and application server technologies, but the implementation should not affect the overall architecture. What described here is the concept of ESB and not the technical infrastructure of how an ESB is implemented.

ESB acts as the intelligent layer for connecting information systems, various data, and other services that are commonly distributed throughout an enterprise IT environment. It combines its core synchronous and asynchronous messaging backbone with intelligent transformation and routing capabilities, and it ensures that messages are passed reliably. ESB enables the intelligent processing of service requests and responses but also events, and messages which is much more interesting when discussing EDA in the next chapter. (Bieberstein et al 2006)

ESB is not a new product as such, but a new concept to integrate information systems, coordinate resources, and manage information. Unlike many previous approaches for connecting information systems, ESB enables the connection of software that runs on different platforms, is written in different programming languages, and uses different technology.

4.8 Business process Management (BPM) and Business Activity Monitoring (BAM)

When implementing and applying SOA it is significant to understand both BPM and BAM concepts. We will describe both concepts in this chapter and will not be further described in chapter 5 discussing EDA. Business Process Management (BPM) is another step in further aligning business and IT. Campbell and Mohun (2007) are looking at BPM as a mean for bridging business and IT and consider SOA and BPM as complementary. SOA is really about enabling agility within IT through loosely coupling software components and standardizing integration through new technology protocols and advanced metadata. You can think of SOA as the ideal enabler for BPM. BPM tools focus on the design of the business process. They model the business process both in terms of what various applications are expected to do and what the human participants in the business process are expected to do. In the following figure you see different IT capabilities coming together to enable BPM.

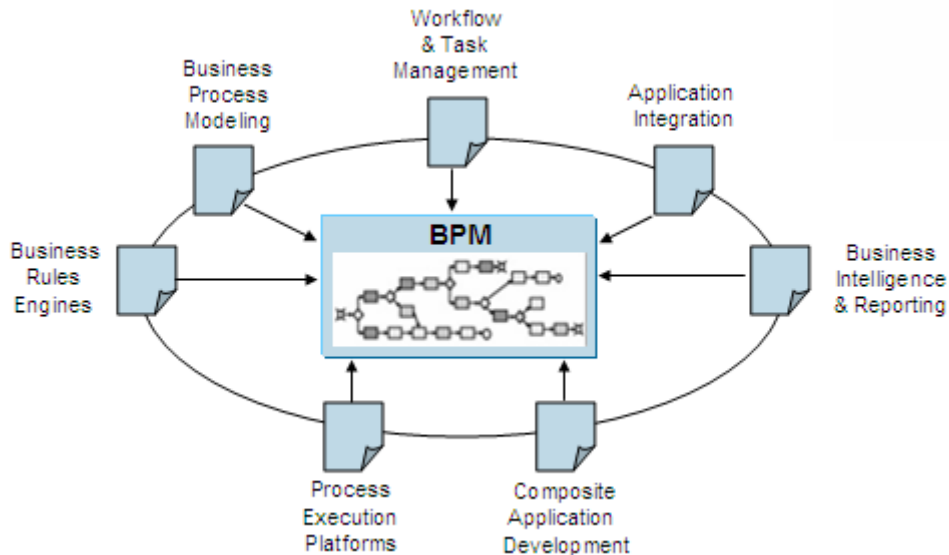


Figure 12 Different IT capabilities enabling BPM (Campbell and Mohun 2007)

The vision of BPM, enabled by SOA, is to package IT capabilities as a set of reusable and repeatable services. Orchestration of business processes using services provides flexibility in the business process to twist and change on demand. The enterprise has gained time to sense and respond to changes in the market place. (Campbell and Mohun 2007)

When having the business process in place a need is often raised to get better insight in the day to day operations. One way of achieving control of day to day operations is by using Business Activity Monitoring (BAM). BAM is often described as process-driven business intelligence, which almost provides real-time access to critical business data to improve the speed and effectiveness of business operations. Most of the input for BAM is event data generated within the past few seconds or minutes, but generally less than an hour. Most BAM systems run continuously, listening to incoming activities and communicating with business people through automatically updated dashboards, e-mail or other channels. BAM systems typically combine event-driven (push) and request-driven (pull) communication to interact with business people. The event-driven communication alerts people when something significant has happened. The request-driven communication enables users to ask the system to drill down into root causes or look up information needed to formulate a response. (Schulte and Gassman 2009)

4.9 Web Services

Even though SOA as an architecture concept is implementation neutral, its association with Web services has become usual, so much that the primary SOA vendors have shaped their respective platforms around the utilization of Web services technology. For that reason we will only discuss Web Services within this chapter. The most significant aspect of Web Services is that every software and hardware company in the world has positioned itself around these technologies for interoperability. Web services allow systems to communicate with each other using standard Internet technologies. Systems that have to communicate with other systems use communication protocols and the data formats that both systems understand. The problem with these communication technologies is that not every platform supports them. Developers must create gateways to convert an unsupported protocol and data format into one that the target platform understands. The emergence of the Internet has forced vendors to support standards such as HTTP and XML. Over the past few years, vendors and

their customers quickly realized that programs that communicate with each other could also use the technologies that run the Internet. Web Services use Internet technology for system interoperability. The advantage that Web Services have over previous interoperability attempts, such as CORBA, is that they build on the existing infrastructure of the Internet and are supported by virtually every technology vendor in existence. As a result Web services are platform-independent. This means that whether the Web service is built using .NET or J2EE, the client uses the service in the exact same way. Below is a list of other benefits using Web Services. (McGovern et al 2003)

Reusability - web services can wrap legacy applications, databases, objects, and components and expose them as reusable services.

Location transparency - a service environment achieves location transparency, because the location is stored in a registry. A client finds and binds to a service and does not care where the service is located. Therefore, an organization has the flexibility to move services to different machines or to move a service to an external provider.

Composition - developers assemble applications from a catalog of reusable services. Services do not depend on the business processes into which they are composed. Because services are independent, developers will logically reuse these services in many business processes.

Scalability - a system is scalable if the overhead required to add more computing power is less than the benefit the additional power provides. Because service clients know only about the service interface and not its implementation, changing the implementation to be more scalable and available requires little overhead.

Reduced vendor dependence - as long as the platform used to build the application supports Web services standards, it is irrelevant to the consumer of the service. Web services allow organizations to make decisions about which platform to use based on the merits of the platform rather than vendor lock-in.

4.10 Summary

SOA is not a product category or a technology platform. SOA is an architecture concept containing many aspects of an enterprise including business processes, information management, information system architecture, and supporting technology platform. Business service is the DNA and core component within SOA. All assets and resources within an enterprise are grouped within different services which have well defined boundaries, delivering specific value and functionality.

Business and technology services are following specific principles making them compatible within service oriented architecture. The services are delivering business functionality and can be put together to support business processes. It is claimed that due to the nature of the services and applied principles a service oriented architecture is expected to be flexible and dynamic to change according to both internal and external needs.

SOA is not a big bang implementation model. SOA is achieved incrementally through time, at project level by continuously defining and enforcing the standards that it will be based on. The standards are the policies that in the aggregate define your SOA conceptual architecture and, when implemented, help your organization achieve its SOA vision and business goals.

5 Event-Driven Architecture (EDA)

In this chapter the main characteristics of Event-Driven Architecture are covered. Including both objectives of this concept and its building blocks.

5.1 Background

Event driven software architecture and design patterns have been around and implemented for many years. It has mostly been used to design and implement applications with high performance requirements handling thousands of transactions. During the last couple of years event driven architecture has had its renaissance much due to the entry of SOA. The relation between SOA and EDA has been a hot discussion topic since that. People and researchers are of different opinion how Services and Events should interact at different levels including business, information, and information systems. Another interesting discussion going on is if EDA is an architecture style or just another way of implementing SOA? Also the technical infrastructure is a valid topic for studies since several of the major infrastructure suppliers are providing platforms supporting both SOA and EDA, or at least pretend to!

In this chapter EDA will be studied and described. We will not only describe EDA from an information system architecture perspective but also from an enterprise architecture perspective and how EDA will be help to support business needs from an IT management standpoint.

5.2 Characteristics of EDA

Much like SOA, Event-Driven Architecture is trying to achieve and increase agility and flexibility within organizations. Before describing the architecture the characteristics of an event-driven company are summarized in this section.

According to Ranadivé (1999) the event-driven state of mind has been used for many years. The event-driven company's core strategy is to maintain competitive flexibility by structuring itself for immediate response to business events in its surrounding environment, modifying its organization and operations in real time to give customers exactly what they want. In such organizations planning becomes a dynamic, interactive, real-time process guided by long-term objectives and strategies executed in real time. Ranadivé (1999) also points out the ability for information to trigger productive responses within organizations as the central theme of the event-driven company. Being event-driven is more than a technological characteristic, it is the infrastructure, culture, and mindset that is required for companies to stay competitive today and in the future. Becoming event-driven is pointed out as the most powerful tool to achieving a sustainable competitive advantage in today's global business ecosystem! However before choosing the event-driven route you must be certain that it is the right route for your business. The following list is a summary of the most typical benefits EDA will bring to business and IT:

- Event-driven architecture brings you, your customer, and all those operating in your marketplace together in real-time, without barriers of distance or technology.
- By converting information from the passive request/reply paradigm to proactive publish/subscribe, the event-driven company provides each employee, partner, and/or

customer with a comprehensive set of real-time, constantly upgraded, and custom-designed information.

- The event-driven company encourages value-added and active information sharing.
- An event-driven company is learning from its mistakes. In an event-driven infrastructure it is impossible to sweep anything under the rug and pretending “you don’t want to hear about it”, “you don’t want to look”, or “you may have missed it”. The company will be aware of what is going wrong the moment it starts to go wrong.
- The event-driven technology infrastructure is based on opening and integrating closed and proprietary information. (Ranadivé 1999)

5.3 Concept

Event Driven Architecture is an architecture style characterized by the existence of a number of relatively independent actors who communicate events amongst themselves in order to achieve a coordinated goal. The event-driven architecture features a technology called “publish/subscribe,” which allows information about business events to be distributed in real time. EDA is based on the asynchronous publish-and-subscribe pattern, where the publisher is completely unaware of the subscriber and vice versa. Services are loosely coupled in the sense that they only share the semantics of the message. (Ranadivé 1999) While SOA is generally a better fit for a request/response exchange, EDA introduces asynchronous capabilities. In an event-driven architecture nodes are generating events and does not depend on the availability of the receiving node. It is really decoupled from the other nodes. (Marechaux 2006) EDA is usually implemented as a type of SOA, stressing the use of fully asynchronous, one-way communication patterns, rather than the more common SOA communication patterns, such as request/reply. (Schulte and Sholler 2008)

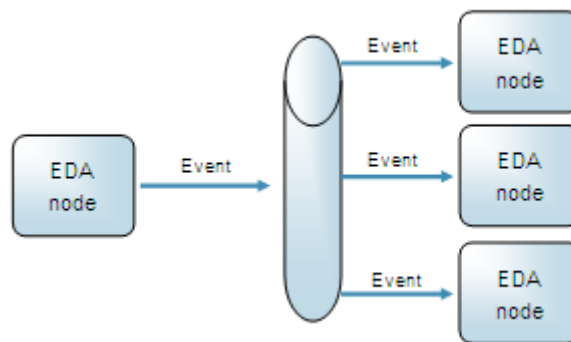


Figure 13 The publish/subscribe mechanism in an Event-Driven Architecture

When seeking to support independence between business processes steps, EDA can provide significant benefit. In an event-driven architecture when a notable thing happens inside or outside your business, it is spread immediately to all interested parties (human or automated). The interested parties evaluate the event, and optionally take action. The event-driven action may include invocation of another service, and/or further information publication, and/or new event creation. By its nature event-driven architecture is very loosely coupled. The creator (source) of the event only knows the event created. The creator has no knowledge of the event’s subsequent processing, or the interested parties. Event-driven architectures are best used for asynchronous flows of work and information. (Michelson 2006)

The term event is often used to refer to both the specification (definition) of the event, and each individual occurrence (instance) of the event. For an event to be meaningful to subscribers (human and automated) it is essential that the event (name and body) is specified in business terms, not data or information system terms. Another definition of event is brought from Wikipedia.

“An event can be defined as a significant change in state. For example, when a consumer purchases a car, the car's state changes from "for sale" to "sold". A car dealer's system architecture may treat this state change as an event to be detected, produced, published and consumed by various applications within the architecture.” (Wikipedia, the free encyclopedia, 12th December 2009)

Hoof (2007) has summarized business event definition process in several steps. The event definition process should start by recognizing the business events that your business is planned to react upon. At this level the IT-implementation is not considered, and the event definition should be in focus. The initial event definition may be figured out from what you currently need to know to appropriately react on the event. Joshua Oyugi in the interview from 9 July 2009 is also referring to the significance of event definition and uniqueness of an event when implementing EDA. He also mentions the use of an “event dictionary” which will facilitate the definition and governance of events.

Next step is to define the process that detects the event. Here comes the producing service(s) to the scene. A crucial factor in designing the events is that it shall be designed in such a way that it not only fits the current requirement, but also future requirements. This can be accomplished by focusing on the characteristics of the event itself when defining the event, and not the current known consuming services or business process.

When defining the event-producing and –consuming services you are diving into a more detailed layer. Each event occurrence has an event header and event body. The event header contains elements describing the event occurrence, such as the event specification ID, event type, event name, event timestamp, event occurrence number, and event creator. These elements are consistent, across event specification. (Michelson 2006)

The event body describes what happened. For example, if a retailer specified a low inventory threshold event, the event body would contain the information to communicate which product fell below the allowable threshold. The event body must be fully described so any interested party can use the information without having to go back to the source system. For the low inventory threshold event, the event body would contain not only the product identifier, but also the product description, and the point in time inventory and threshold levels. To ensure events are understood by all consumers, a clear business lexicon or ontology should be used. (Michelson 2006)

5.4 Principles

EDA principles are not listed and discussed as extensively as SOA principles. The reason for this may be that it has not been in focus as much as SOA during the last years and the maturity level of the existing literature and technology is not at same level as SOA. The list below is a summary of the main principles which have been encountered during literature study and interviews. This is an area requiring more research. We predict as the technology is

developing to support EDA the focus on EDA will increase and main design principles defined.

Decoupled – The creator (source) of the event only knows the event created. The creator has no knowledge of the event's subsequent processing, or the interested parties. (Michelson 2006) This principle will assist in both creating decoupled application integrations and business processes. A process is composed of multiple stages. In an EDA the stages have no physical dependencies and a minimum of logical dependencies on each other, so each can be modified without causing side effects on the others, as long as the notification messages do not change. (Schulte and Sholler 2008)

Lennart Eriksson (interview 3 September 2008) and Joshua Oyugi (interview 9 July 2009) are also referring to this as one of the main EDA principles where event producers and consumers are not aware of each other.

Reusability – A crucial factor in designing the events is that it shall be designed in such a way that it not only fits the current requirement, but also future requirements and out-of-scope usage. Events must be designed in a generic way which is able to reuse. (Hoof 2007)

Real-time notification – The technical infrastructure must support real-time creation and delivery of events. The human infrastructure must support real-time transformation of information first into knowledge and then into intelligent and following action. (Ranadivé 1999)

Publish/Subscribe – Notifications are pushed by the event source, not pulled by the event consumer. The event producer determines when the message that contains the event is sent. (Schulte 2008)

Immediate response – The arrival of a notification causes the event consumer to act immediately. However in some cases, the action is simply to save the event for subsequent processing. (Schulte 2008)

Freedom to act – A notification does not specify what action the event consumer will perform. It is a report not a request. The consumer contains the logic that determines how it will respond. (Schulte 2008)

This principle is also referred to by Lennart Eriksson in his interview (3 September 2008). According to him a huge number of events may be produced but only a few will be consumed by the consumers.

Statelessness of event processing components – The event processing components are stateless but the event itself will keep the state defined within it. (Joshua Oyugi, interview 09 July 2009)

5.5 Data

Ranadivé (1999) stresses the importance of information in an event-driven architecture. An event-driven company instantly senses and responds to the events that drive its business and uses the power of information to drive the development of new products and services. In such a company business information is distributed in real time to those in an organization whose

optimal functioning depends upon it. Once information is delivered via publish/subscribe, it should begin the enrichment loop, inserting itself into applications that analyze and refine it and then send it to all interested parties.

Data distribution and loose coupling also put some requirements on the data in an EDA. Loose coupling means independency where components do not rely on each other. Not even on each other's stored data. Each loosely coupled environment maintains its own copy of the data. This may lead to data redundancy. In loosely coupled environments redundancy must not be seen as poor design, but as strong design. Banning redundancy across decoupling borders makes the coupling more tightly. Maintaining redundancy across the decoupling borders (synchronization) makes the loose coupling more robust. (Hoof 2007)

EDA does also stress the importance of commonly understood semantics. Shared semantics is a prerequisite in connecting information systems, no matter whether it concerns EDA, SOA or any other form of enterprise application integration (EAI). It should be obvious to anyone that analysis of data semantics will always be the first activity of any integration project.

In an EDA business event is represented in a standardized format with decided semantics. This format and semantics are defined in the enterprises Meta data model or sometimes called Canonical Data Model (CMD). The CMD is not a storage component, but a metadata component. The CMD holds definitions of the local formats and semantics of the participating systems and the definitions of the canonical or standardized format. The transformation service provided by e.g. ESB will take care of the transformation between the local and canonical format. In this way everybody can use their own model, formats and semantics.

5.6 EDA implementation components

So far we have touched several of the implementation components required for an event-driven architecture. Further has Michelson (2006) described the required EDA components in a layered architecture (see Appendix C). The components can be broken out in five categories:

Event Metadata - A good event-driven architecture has a strong metadata architecture. Event metadata includes event specifications and event processing rules. Event specifications must be made available to event generators, event format transformers, event processing engines, and subscribers. (Heffner 2006)

Event Processing - The core of event processing are the engine and the event occurrence data. Simple event engines are often homegrown while more complex event engines are provided CEP (complex event processing) engine providers.

Event processing includes several services e.g. event generation, event response initiation, and event response processing. Events may be generated because of things that happen or things that don't happen within a specified time period. Examples of potential event (and non-event) sources include portals, business services, infrastructure management tools, BPM infrastructure, RFID endpoints, integrations servers, databases, and many more. Event processing infrastructure, like complex event processing (CEP), correlates and analyzes event flows and initiates event responses which may be either automated responses or workflow items for human review and processing.

Event Tooling - Event development tools are required to define event specifications and processing rules, and to manage subscriptions. Event management tools provide administration and monitoring of the event processing infrastructure, monitoring of event flows, and visibility into event generation and processing statistics.

Enterprise Integration - An enterprise integration infrastructure plays a large role in event-driven architecture to filter, route, transform, transport, invoke services, invoke business processes, etc.

Sources and Targets - These are the resources of the enterprise applications, services, business processes, data stores, people, and automated agents that generate events and/or perform an event-driven action.

5.7 Complex Event Processing (CEP)

Simple event-driven processing has been in common use for at least 10 years with technology such as IBM's or Tibco Software Inc.'s message-oriented middleware and, in the past few years, message-driven Enterprise JavaBeans. But complex event processing (CEP) is becoming the mainstream since 2007, as architects and business analysts strive to do more business in real time. (Sliwa 2003)

“CEP, is primarily an event processing concept that deals with the task of processing multiple events from an event cloud with the goal of identifying the meaningful events within the event cloud. CEP employs techniques such as detection of complex patterns of many events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes.”
(Wikipedia, the free encyclopedia, 12th December 2009)

CEP helps automate intelligent responses to different events. A CEP system identifies notable events, gains an understanding of what the event is, tracks it, and fits it into a larger pattern so action can be taken. While SOA and orchestration automate the processes, CEP automates the intelligent correlation and decision-making done by individuals. A CEP system allows users to organize random, unrelated events, find trends, and to predict outcomes. Action can then be taken to prevent a negative outcome from occurring. The patterns can also be analyzed to improve the underlying processes and information systems to prevent future mistakes.

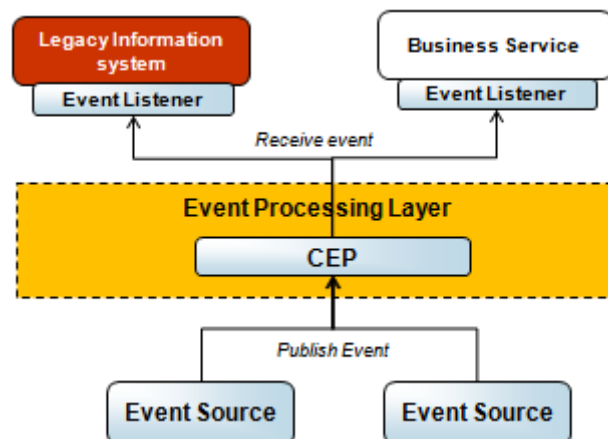


Figure 14 CEP function within EDA

5.8 Summary

EDA is another architectural style which just like SOA is trying to shape future organizations and enterprises to be agile and adaptive to internal and external changes. The key message of EDA is to support organizations in creating an adaptive and proactive environment where real-time information is communicated to all interested parties. The importance of information in event-driven architecture is obvious as the “event-driven enterprise” as a company that acquires, deploys, and wisely exploits real-time, active information.

Except the important role of information the business event itself is the DNA of EDA. The business event is the information carrier. EDA is a business event focused design approach by recognizing the events that your business planned to react upon. Business events are instantly and automatically delivered to everyone in the company.

Event-driven architecture applies to specific design principles and requires some specific infrastructure services such as EAI, CEP and Metadata repository. EDA supports a decoupled architecture where services are unaware of each other’s existence. This creates an adaptive and flexible architecture. Though EDA has been in use for many years there is an obvious gap in supportive design principles, standards, infrastructure, and design tools.

6. Analysis

In this chapter SOA and EDA are compared from different views and aspects. The comparative analysis describes the differences and similarities between the two architecture styles and how they interact and affect each other in a common environment.

6.1 Comparative Analysis

During this study it has not been able to find an obvious comparative analysis of the two architecture concepts. The literature is often describing these two styles as complementary, analyzing benefits and disadvantages of each one. Though we also agree that SOA and EDA are much complementary we find it necessary and useful to do a comparative analysis discussing different aspects of the two styles. The reason for that is variety of the principles that need to be clarified and emphasized. This comparative analysis is divided into three different categories (business, information, and information system) which can be traced back to Enterprise Architecture framework. Technology is one additional category within Enterprise Architecture which has been excluded from the scope of this thesis.

6.1.1 View on Business

In this section we are analyzing and discussing how SOA and EDA are considering business function and business processes. Both architecture styles are aiming at increasing business flexibility and agility. Both SOA and EDA are focusing on making business processes and functions more dynamic to react and adapt to new changes and challenges. For business, SOA and EDA means increased customer satisfaction, real business agility, faster time to market, better business intelligence, and lower business cost at the end.

The main difference between the two architecture styles and their view on the business function is the DNA of the business. SOA is focusing on the decomposition of business functions and how they can be reused. SOA is a business concept, an idea or approach, of how IT functionality can be planned, designed, and delivered as modular business services to achieve specific business benefits. Service orientation provides the ability to loosely couple

applications, trading partners, and organizations and to invoke them via service calls. Furthermore independent services can be composed in processes to provide even greater value and automate and integrate business processes. SOA is about recognizing the functions that your business is supposed to use and deliver. As an example a business service can be “registering new customer”, “creating new bank account”, or “updating account balance”. These business services can all be reused to support business functions and processes.

The DNA of EDA is however business event which is a change within the state of an activity. Business events are the cornerstones within EDA providing the ability to get necessary information in real time and be able to react and take necessary actions. An Event-Driven organization is encouraging information sharing helping you to turn your organization into a learning organization and making your business proactive compared to SOA which tends to be more reactive promoting request/reply approach. As an example a business event can be “new order placed”, “order canceled”, or “price list updated”. All these business events are generated by changes within business context and environment which have to be handled.

Both SOA and EDA are aiming at aligning business and IT but by using different means. While SOA is using business service EDA is using business events to bridge the gap between business and IT.

SOA	EDA
<ul style="list-style-type: none"> ○ SOA is focusing on the decomposition of business functions which can be reused ○ SOA is an architectural style that recognizes services (functionality representing process steps) ○ Encouraging reuse of functions (reactive) ○ SOA facilitates business process automation and integration 	<ul style="list-style-type: none"> ○ EDA is focusing on identifying business events which is a change within the state of an enterprise ○ EDA is an architectural style that recognizes events (messages representing process states) that your business is planned to react upon ○ Encourages information sharing and use of business intelligence (proactive)

Figure 15 Comparative Analysis – Business process and function

6.1.2 View on Information

Systems that pass data to each other share commonly understood semantics. Data semantics is the key to success in both SOA and EDA. Shared semantics is a prerequisite in connecting information systems, no matter whether it concerns EDA or SOA. Both styles are considering Information as a common asset and advocates use of a Canonical Data Model which helps the organization to understand and interpret information in a common way. Though both SOA and EDA are emphasizing the importance of data integration and shared semantics there are many aspects of data that are not clearly defined within both concepts.

As an example both EDA and SOA are including structured data and not mentioning unstructured data. Neither SOA nor EDA are explicitly refereeing to structured data but due to the content of the discussion and importance of shared semantics one can interpret the data content as including structured data. It is not either clearly defined if there is a distinction between local or global/common data.

SOA begins with the goal of achieving consistency between data sources. In order to achieve data consistency, you begin by separating your data from its tight dependency on the business applications that created it and update it. Information is managed as a service which is requested by the ones who are interested in that information. SOA is paying a lot of attention to information sharing in a more reactive request/reply way compared to EDA. SOA is providing the ability to identify and access right information at the right time independent on the technology used.

While SOA is much about pulling the needed information EDA is focusing on pushing right information to right people at the right time. EDA is promoting instant identification and responding to the event/information that drives the business. Information is distributed in real-time to the ones who are interested in that information. Compared to SOA, EDA is using information in a more proactive manner encouraging real-time information sharing. Learning organizations should benefit from EDA by using real-time information to gain competitive advantages. To create a learning-organization the need of fresh business intelligence is obvious. EDA does also provide significant support in this are by facilitating real-time information sharing and complex processing and analyzing of events and information.

Another issue to compare is how the different styles are approaching data redundancy. In all traditional architecture styles data redundancy is something which should be avoided. Within SOA data redundancy is not promoted nor declined though one single source of data is recommended to provide information services with relevant data (master data). It has not been able to come across how SOA takes a position on managing data redundancy and if it should benefit or constrain a Service-Oriented organization. However within EDA data redundancy may be necessary to achieve the goals of a proactive organization which is totally decoupled.

SOA	EDA
<ul style="list-style-type: none"> ○ Promotes reuse and share of data in multiple applications and for multiple end-user access channels ○ Information as a service is an architectural approach that loosens the tight connections between data and applications so that data can be controlled and shared across the enterprise ○ Information is requested by the ones who are interested in that information ○ Data redundancy is not promoted nor declined ○ Use of Canonical Data Model ○ Master data provides “information services” with data ○ Includes structured data 	<ul style="list-style-type: none"> ○ Promotes instant identification and responding to the events/information that drive the business ○ Information is distributed in real-time to the ones who are interested in that information ○ Uses the power of information to drive the development ○ Data redundancy is a necessity to increase decoupling ○ Use of Canonical Data Model ○ Includes structured data

Figure 16 Comparative Analysis – Information

6.1.3 View on Information System

Within SOA information systems are modeled as service providers and service consumers. The information system architecture is based on modularized components and services which are increasing the flexibility and ability to change business processes and minimize time to market. Within SOA information systems are aware of each other and the functionality each system is providing. This approach is requiring a request/reply mechanism where the service consumer is asking the service provider. While SOA is focusing on information system's ability to provide and use services EDA is focusing on events triggering messages. The messages are sent between independent information systems or software modules that are completely unaware of each other.

Within EDA information systems are providing or consuming events and must decide what actions to carry out and/or what events to generate.

SOA	EDA
<ul style="list-style-type: none"> ○ Systems are modeled as Service Providers and Service Consumers ○ Applies incremental development and maintenance of large distributed applications 	<ul style="list-style-type: none"> ○ An approach for designing and building systems in which events trigger messages to be sent between independent software modules that are completely unaware of each other ○ Applies incremental development and maintenance of large distributed applications

Figure 17 Comparative Analysis - Information System

6.1.4 View on Integration

Another interesting area to discuss is how the two architecture styles are dealing with integration. SOA is often mentioned when discussing information system integration. One reason for that is how SOA is changing the need of integration focusing on business services instead of information systems. SOA promotes integration by separating steps of wrapping, layering and composing supporting both synchronous and asynchronous patterns. Loose coupling is one of the guiding principles within SOA and is provided because the Service Providers and Service Consumers use Service Definition as an interaction contract and services are invoked independently of their technology and location. While SOA is refereeing to "loose coupling" as a guiding principle EDA is tackling integration in a more "decoupled" way where information systems are not aware of each other. Within EDA information systems are integrated through events that broadcasts state changes. Other information systems or services that are interested in these events are subscribing to these events and choose to respond to it or not. Messages are typically sent using the publish/subscribe approach because it enables simultaneous delivery of messages to multiple destinations. Due to the fact of the publish/subscribe mechanism EDA is forced to use asynchronous pattern.

SOA	EDA
<ul style="list-style-type: none"> ○ A SOA-based application consists of business components that supply services and other programs that act as clients, or "consumers," of those services 	<ul style="list-style-type: none"> ○ Information system or services broadcasts State changes using Events. Other systems that are interested in these Events can subscribe to these Events and choose

<ul style="list-style-type: none"> ○ Promotes integration by introducing standards and separating steps of wrapping, layering and composing ○ Request/Reply pattern ○ Can use both synchronous and asynchronous patterns ○ Loose coupling is provided for because the Service Providers and Service Consumers use the Service Definition as a communication and interaction contract ○ Services are invoked independently of their technology and location ○ One specific service is invoked by one consumer at a time (one-to-one communication) 	<p>to respond to it</p> <ul style="list-style-type: none"> ○ Prescribes asynchronous pattern ○ Decoupling is provided since event publishers are not aware of the existence of event subscribers ○ Publish/Subscribe messaging where one specific event can impact many subscribers (many-to-many communication)
---	---

Figure 18 Comparative Analysis - Integration

7 Discussion

The relation of SOA and EDA has been discussed during the last five to six years where people are of different opinion. Ranging from EDA being the “new SOA”, to EDA “succeeding” SOA, to EDA “extending” SOA, to pure skepticism of any relationship at all. In this chapter the relation between EDA and SOA and the interacting touch points are summarized.

7.1 Business Objectives

Both SOA and EDA move organizations from an old architecture based on processes supported by independent monolithic information systems tightly coupled together, to a new type of architecture based on independent services and events that are more dynamic and focusing on reusable suites. Both concepts as they are described in this study are long-term investments leading to several areas where they may bring value e.g. by

- wrapping existing information systems as services the life and ROI of legacy can be extended and reducing the cost of development of new functionality
- leveraging existing services and events new business processes support can be deployed more quickly enabling faster and lower cost to market
- reusing common services and events within several business processes, units or enterprises will lead to a better Return on Investment (ROI)

Both SOA and EDA promise to add value in a frequent changing and dynamic business environment. They bring value to business by making business processes and supporting IT environment more flexible to adapt to business challenges. Based on this discussion we can summarize that a dynamic business environment undergoing frequent changes is a prerequisite when deciding to implement SOA and EDA. However most often all parts of a business will not undergo same level of change leading us to the conclusion that SOA and EDA does not necessarily need to cover and get implemented in all parts of a business. Most often the majority of the functionality provided in a organization is not provided in the form of services, and it is unrealistic to think that. It is the responsibility of the architect to discover the most relevant and critical parts of the business which may be within the scope of a SOA or EDA implementation. It is also the responsibility of the architect to use legacy functionality into business processes just as effectively as with new designed business services and events.

Another important aspect is how to reach, harvest, and value the promised benefits by implementing SOA and EDA in combination. There are increasing pressure to increase the Return on Investment (ROI) from IT projects. The promise of both SOA and EDA is that reuse will cut IT costs. They also promise the potential for implementing business processes in a shorter period of time. The issue is however that business services and events are parts of business processes. They involve people and information as well as functionality. In defining SOA and EDA you are designing and organizing business processes and organizations as well. If services and events are to be reused they must fit well into existing business organization and multiple business processes. This fact adds additional complexity to EDA and SOA projects and may in many cases lead to a lower ROI due to deminished level of reuse. SOA and EDA also face challenges in existing silo-base projects where business services and events are identified and designed for the need of a single business unit or process. These services will not fit into other business processes and must be redesigned or new services must be created. A multiple business process and business unit perspective may

not exist and not fit into traditional silo-oriented IT projects. The silo implementation of business services and events generate a higher initial cost and will not lead to any benefits until the reuse actually occurs.

7.2 Enterprise Architecture context

The main objective of Enterprise Architecture is to align business and IT initiatives and support IT management efforts in a structured manner. During the years different Enterprise Architecture frameworks have been developed to support architects in planning and structuring IT management efforts. The frameworks have also had major impact on deciding the content of Enterprise Architecture. Unified Architecture Framework (UAF) is one of many frameworks including several architecture areas. Both SOA and EDA can be mapped into UAF. The first architecture area in UAF is “business”. Both SOA and EDA as architecture concepts are strongly emphasizing the need of a strong business architecture where business needs are well understood and business processes are defined and broken into well defined services and steps that can be rearranged and reused. What is less obvious when looking into business architecture is the granularity level of business services and business events. Defining business services and events with right level of granularity is critical for the success of both SOA and EDA. So far we have not been able to identify a method for how this can be managed.

Information architecture is another architecture area. Information is central to business processes. Business processes determine what information is needed and how they should be managed. Both SOA and EDA illustrate the importance of information and advocate a single common data model and common business language. This is one of the corner stones in both SOA and EDA. EDA is also mentioning data redundancy as a design pattern to increase decoupling. Though we understand the logic behind use of data redundancy to increase decoupling we believe that the need of it has to be analyzed case by case. Architects have to understand and evaluate benefits of implementing data redundancy and how it may impact the journey towards a single common data model. Another aspect of information architecture is availability of information. Both SOA and EDA strive for making right information available using information services which is highly dependent of a common data model. While business services are responsible for managing and retrieving information business event are responsible for transferring that information. The challenge when combining SOA and EDA is that services and events must be designed at the same granularity level. This will add to the complexity of an architecture supporting both EDA and SOA.

Information System is the third architecture area in UAF. This architecture area is however less comparable with any architecture layer in SOA and EDA. Traditionally Information Systems have delivered a specific amount of functionality in a structured and predefined approach. Both Service-Oriented and Event-Driven Architecture are new in the sense that they are dependent on and advocating independent services that will cooperate to deliver functionality. In this scenario the role of Information System as such is less clear! The question “what are the Information Systems in an environment applying SOA and EDA?” is much relevant. In an environment where EDA is implemented the question will be even more relevant since EDA is promoting even greater independency and higher level of decoupling between services and events. It is also important to mention that Information Systems go beyond providing functionality. When a business process deploys services it must be decided when and how each service is deployed. This is called service orchestration in the SOA world. During this study we have not been able to come across any specific answer to this question.

However what is clear is that both SOA and EDA consist of smaller services and information packages that dynamically can be arranged and rearranged to serve different business needs. This is an environment where definition of Information System is not straight.

The lack of a clear “Information System” does also impact other IT-management issues such as governance. Who is responsible for each service or event in an organization? Traditionally in many organizations people have been responsible for one or several Information Systems. Due to the lack of a clear Information System in SOA a different mindset is required to govern and control the IT environment. Integration is another area which needs a different approach. Architects need to track and maintain integrations between services and events instead of Information Systems. As the level of flexibility is increasing applying SOA and EDA it is proven that the need of governance and control is increasing. This is both due to splitting up an Information System into services and events but also due increased rate of change that services will undergo compared with a monolithic Information System.

Technology Infrastructure is the forth architecture area in UAF. This architecture area has been left out and not covered in this thesis. However what is worth mentioning is that both SOA and EDA are dependent on new infrastructure services and capabilities. In some cases the same infrastructure capabilities may support both SOA and EDA.

7.3 Architectural relations and dependencies

Despite many similarities main differentiators between the two styles are mainly within architecture objectives and building blocks. The architecture objectives can be summarized as follows.

SOA architecture objectives	EDA architecture objectives
<ul style="list-style-type: none"> ○ Support ease in rearranging business services to create a higher level of flexibility within the business and IT environment ○ Identifying well defined, modularized business services with clear responsibility ○ Loose coupling between business services ○ Reuse of business services 	<ul style="list-style-type: none"> ○ Support ease in rearranging the events to create a higher level of flexibility within the business and IT environment ○ Creating a more proactive business environment ○ Supporting Real-Time information sharing ○ Supporting a learning organization by detecting patterns in sets of events ○ Decoupling ○ Reuse of business events

By comparing architecture objectives listed in the table above we can conclude that there are no conflicting goals and we are able to identify three categories of relations. (1) In the first category both EDA and SOA are sharing the same objectives. Two main goals which are shared by both EDA and SOA are to support ease in rearranging components and supporting concurrent use of components in different contexts. Note that these are commonly accepted with regard to services, but are only recently mentioned in the context of events. (2) In the second category EDA is extending SOA. The two most obvious examples are how EDA moves SOA from being loose coupled to a decoupled architecture and how EDA supports a more proactive business environment. And finally in the (3) third category we see some dependencies between EDA and SOA architecture when they coexist, which is discussed in this section.

After a closer look at services and events within SOA and EDA you can identify a close relation between the two main components (events and services). Two main relations between EDA and SOA are obvious. In the first relation, events connect services by transferring process state and data from one service that detects and publishes events to other services that are triggered by specific events. In the second relation, services connect events by transferring the process from one state to another. In other words the event is holding the state and the service is changing the state.

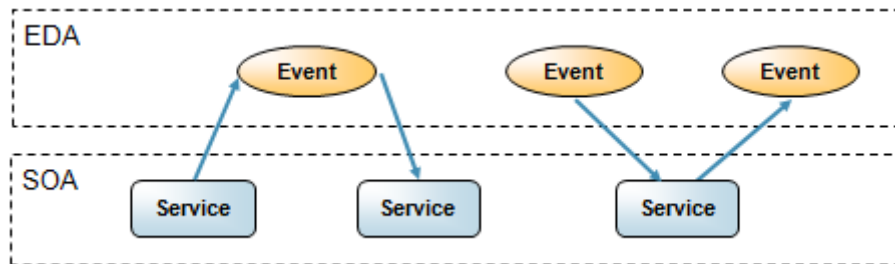


Figure 19 Service and Event relation

As discussed previously in this study one critical issue when analyzing the relations above is the level of granularity to be implemented. By this we mean that in a well defined architecture the business events are defined at a granularity level which is well aligned with the coarse-grained business services. Though the granularity level of both services and events are of great importance it is not clear how to reach right level of granularity during the definition phase. It is even less obvious how to make sure that services and events are at the same level! During this study no guidelines or frameworks have been identified which can be used to support architects during the definition phase of services and events.

Another way of demonstrating the relation between EDA and SOA is by studying architecture layers. During the interview with Joshua Oyugi the architecture layers were discussed.

Joshua Oyugi demonstrates the relation by adding a new “event processing layer” in between traditional SOA layers. The events managed through this layer can be published by services or the orchestration engine which also can consume published events. The new layer is helping us achieving a number of main goals where the first one is decoupling and the second one is real-time information sharing. Another benefit is getting the business process to be more proactive.

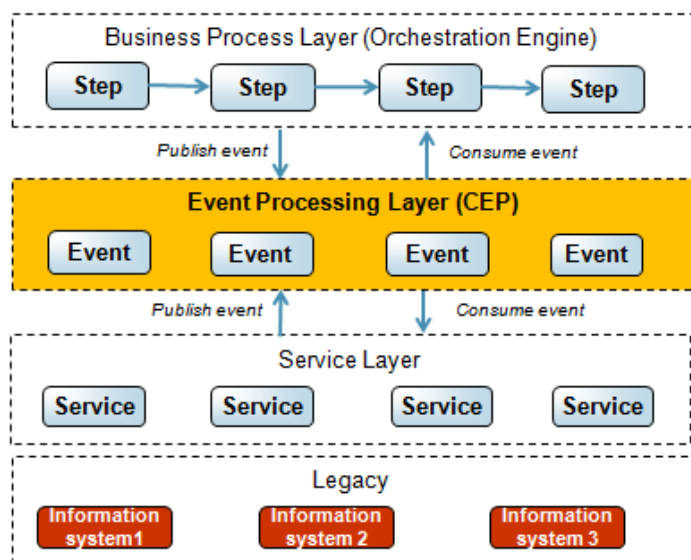


Figure 20 SOA and EDA Architecture Layers

Also here it is not obvious how the service layer and event layer should collaborate and integrate. No design patterns or frameworks have been located during this study describing a best case scenario of service and event integration.

7.4 Service decoupling

SOA is promising loose coupling by using services. However this may not be the only truth since remote services are called relying on their existence, predefined contract and availability of foreign data. This makes the performance of the business processes dependent on external entities. This is a way of tight coupling! Business processes can still be rearranged quickly with reusable building blocks. But on the other hand you may not fully follow the loose coupling principle.

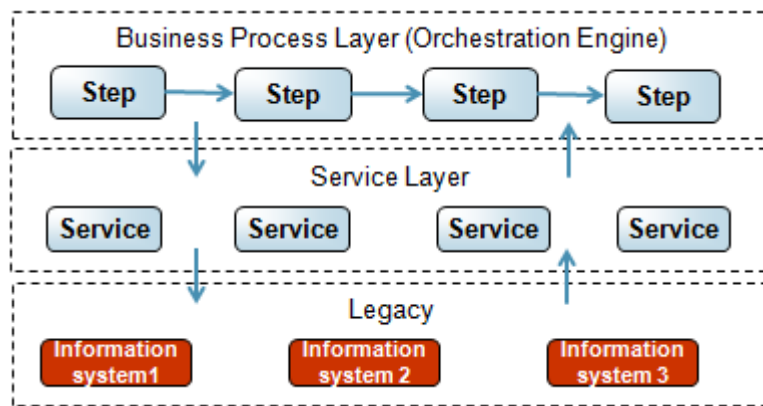


Figure 21 SOA Business Process

Figure 21 visualizes Service-Oriented Architecture where different information systems or partners are wrapped into services and integrated using a request/reply pattern. A better approach at the business process level is to decouple business process using Event-Driven Architecture. EDA is an architecture style in which events trigger the real-time exchange of messages between independent software components. In figure 22 SOA components are no longer directly coupled, but connected via decoupling points (events). Here we start applying the event-driven design principles. It is about reusable data (events) as well as reusable services. The implication of the new architecture is that the initiative for data exchange is not taken by the consuming application, but the producing application takes the initiative. It decouples information systems and so the supported business process. Consuming applications can be added and removed as much as needed without affecting any of the other applications or processes. At the same time the consumer is independent of the availability of the publisher to some degree. Though the level of dependency between services is diminished using events in between the consuming service is still dependent on the publishing service publishing new events. In this scenario some responsibility is taken from the consuming service and put on the publishing service.

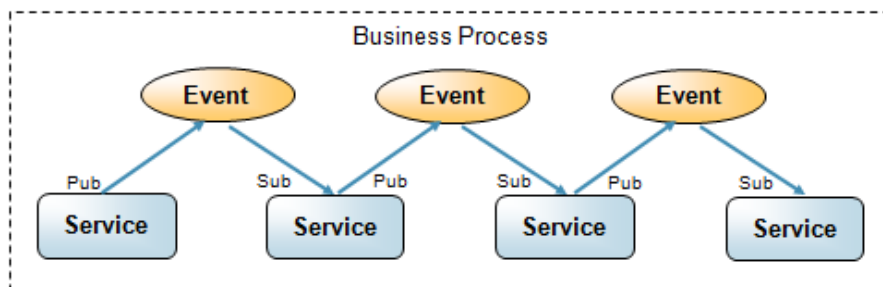


Figure 22 SOA and EDA Interaction

By studying figure 22 we discover the interaction between SOA and EDA which is mainly taking place at two different levels:

1. In the first relation events connect services by transferring process state from one service that detects and publishes events to other services that are triggered by specific events (the publisher which may be a service within the service-oriented architecture generates an event).
2. In the second relation services connect events by transferring the process from one state to another (the subscriber receives the event and takes necessary actions which may lead to calling new services and generating new events).

It is also important to point out that the interaction between SOA and EDA also can take place at information system level and not only at business process level. The next figure is illustrating a combination of SOA and EDA where services are no longer the only consumer of events. External/internal information systems may also consuming and producing events.

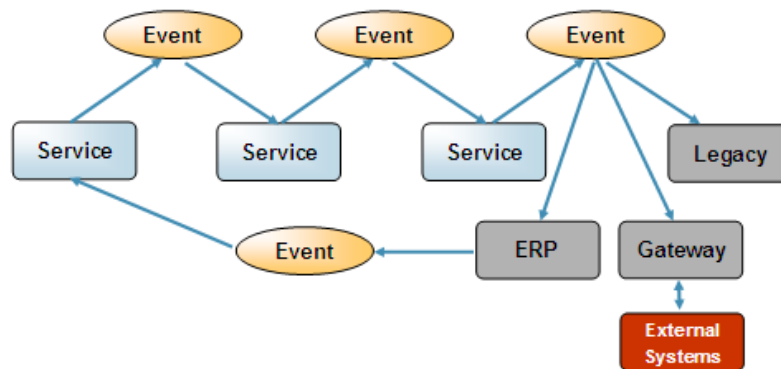


Figure 23 SOA and EDA Architecture

Another way EDA helps you extend and decouple your SOA is by applying data synchronization to be able to handle data redundancy. Data redundancy is something architects mostly try to avoid. However it may be needed in EDA to reach the level of decoupling needed. EDA provides some design principles supporting management of data redundancy and data synchronization. Consider you have three information systems dealing with customer contact information which must be synchronized. All these three information systems are interested in customer contact information and subscribes on events dealing with the right information. Every time a customer contact is modified within any of the information systems an event is generated and published. Subscribing information systems will immediately get the event, get informed about the change and update the information necessary. This process puts also some specific requirements on the need of a Common Data Model since the systems may not use the same data model and semantics. In this scenario business services should be used to both generate and consume the business events. Also the services provided by the ESB which is the heart of a SOA infrastructure should be used to manage the event messaging.

Though it is described how data redundancy can contribute to increase decoupling we believe it is still to be demonstrated if and how data redundancy will add benefit. Managing data redundancy may be a rigorous effort which many organizations try to avoid in most cases. Before putting data redundancy into the list of architecture principles of an organizations one needs to analyze the benefits of it in detail and define how data redundancy management will be supported by an Event-Driven Architecture and infrastructure.

7.5 Real-time information sharing

Real-time information sharing and distribution is another benefit of using EDA. We believe this is one of the most significant reasons why organizations should consider using EDA. An event-driven company instantly senses and responds to the events that drive its business and uses the power of information to drive the development of business. This capability is realized by using a publish/subscribe pattern where information systems, people, and organizations are subscribing on information and events which is pushed out as soon as it is detected and available.

EDA is more efficient than SOA if there are multiple destinations for the same data, because the source sends the event only once. An SOA-based client would have to make successive calls. An apparent scenario where EDA is facilitating real-time information distribution and extending SOA capabilities is when implementing BAM, where activities/events are pushed out to business user's dashboard as soon as an activity/event occurs. Real-time information distribution puts some requirements on both EDA and SOA where commonly understood semantics is perhaps the most important one. Shared semantics is a prerequisite in connecting separate information systems and services, no matter whether it concerns EDA, SOA or any other form of legacy integration. Another requirement is right level of service and event granularity. Aligning the granularity level will help you smooth the flow of information between services and event.

When discussing requirements you also have to consider the design principles which have to be applied. We have identified three principles which are of major importance when realizing real-time information distribution. (1) The first one is *real-time notification* which gives us the technical capabilities to deliver real-time, active information and the human infrastructure to transform that information first into knowledge and then into intelligent and ongoing action. (2) The second principle mentioned earlier is of course *publish/subscribe* pattern where notifications are pushed out. (3) And the last but not least important principle is *immediate response*, where the arrival of a notification causes the consumer to act immediately.

In the following example we illustrate how SOA and EDA together can collaborate to enable information integration and sharing. The example is about how a user will get updated with the latest order information and order status. The relevant order information is stored in several information systems where each one has a different data model and semantics for describing an order and the entities within an order. How can you provide a composite view of the order involving several information systems?

Both SOA and EDA in this example are using the same data source and common data model describing the order which all parties can use and agree upon. The common data model is the business language the enterprise will use to describe an order. The next step is to create a business service which is retrieving data from various information systems and/or partners and presents the updated order information to the user. While the business service is responsible for collecting order information, business events are calling the business event every time relevant order data is updated in any of the information systems.

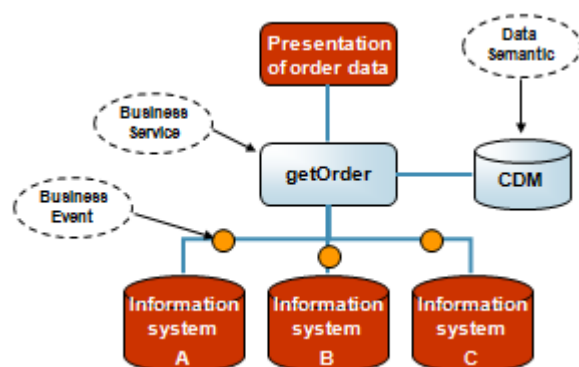


Figure 24 SOA & EDA information handling

The collaboration between EDA and SOA when dealing with information can be summarized in three areas:

1. Both are using the same data source and data semantics
2. EDA is using SOA services to manage the information
3. SOA is using EDA to distribute and publish information

7.6 Common components during implementation

Though technical and physical aspects of SOA and EDA have not been discussed in this thesis it is important to briefly mention ESB as a concept which can be implemented by different technologies. ESB is a major and critical component when implementing both SOA and EDA. The Enterprise Service Bus provides a group of services that are used both when implementing SOA and EDA. These services shall be reused and help to integrate service and events within a business process. ESB will include the business processes, the services, and the events that are consumed or produced.

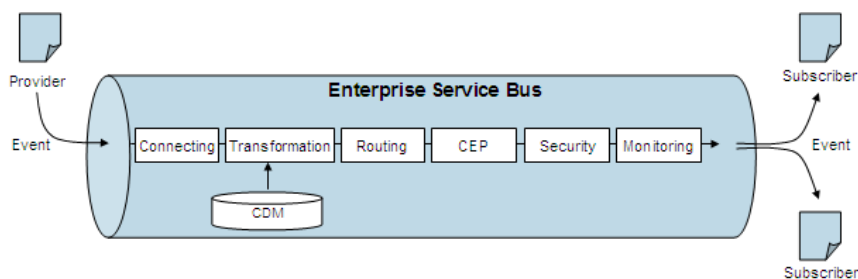


Figure 25 ESB supporting Event Management

Complex Event Processing (CEP) should also be included within the platform to support EDA. We have already described the function of CEP and its role within the architecture. CEP component can be implemented in many ways. In figure 25 CEP is included in ESB as a services.

7.7 Main challenges of combining SOA and EDA

In this paper similarities and differences between SOA and EDA, interaction points and how the two concepts are affecting and complementing each other has already been discussed. It is also necessary to look at the challenges facing an organization when deploying both SOA and EDA. These concepts are new to many organizations requiring changes of mindset and approach both within business and IT. This puts new demands on management group and the way both business and IT are governed and planed. Creating a flexible organization that is promised by EDA is getting more and more challenging in a demanding business environment. By tradition IT applications and processes are geared toward predictable, repeatable events. It's when events happen that are notable or exceptional that most business have trouble. Most companies also find it hard to fit an event into a larger trend that may affect their business. This lead us to the main challenge which is about understanding why and where to implement SOA or/and EDA and what are the main objectives of these initiatives.

It is of great importance to identify parts of the organization which will be benefit from SOA and EDA. This step requires an enterprise architecture definition where the scope of the involved services and events are defined. Business services and events have to be combined in a way enabling flexible reconfiguration of processing, perhaps by enabling multiple responses to an event. All these activities are challenging and require both new competency and C-Level support. Defining services and events and the architecture integrating these is a complex process. During this study we have not been able to identify any guidelines, frameworks or standards which can be used during design of an architecture involving both SOA and EDA. This fact leads to increased complexity during the design and implementation phase. Though both EDA and SOA have been existing for many years we consider the maturity level of an architecture involving both concepts as low.

When studying EDA it is obvious that EDA is aiming at achieving a learning organization that can predict challenges and opportunities and react upon them in a smooth and flexible manner. However during the interviews it was apparent that EDA is still used and seen as a technical integration pattern within or between systems. EDA is perceived as a technical pattern enabling modularity and scalability. The interviewees all agreed that full EDA business benefits are currently not leveraged and that the maturity level is not high. We need to add business perspective to EDA!

Other real-life challenges mentioned during the interviews are governance of EDA events, event definition, security issues regarding events, managing the increasing number of events over time, and interpreting the content of events published². Governance of business events is another area which needs additional research. During this study we have not been able to identify any literature or study focusing on governance of events within an event-driven architecture. Neither SOA nor EDA are discussing this issue and how the increased number and versions of services and events should be managed over time.

Another challenge which also is a key success factor is making EDA part of your SOA initiative from the start. The SOA services and EDA events must be defined from a business perspective and be at the same granularity level. Just developing Web Services and Events without involving the business will not bring the benefits promised by SOA or EDA! Also here we lack guidelines and frameworks supporting architects in defining right services and events at the same granularity level.

² For more details from the interviews see Appendix D

8. Conclusions

This final chapter of the thesis deals with summarizing the content i.e. how EDA and SOA can function within the same context and environment.

The study intends to answer the following problem statement:

How can Event Driven Architecture interact and function with Service Oriented Architecture?

8.1 Interaction between SOA and EDA

By studying the literature, conducting a comparative analysis and discussing architecture objectives it became evident that there are several areas where EDA and SOA are interacting. People have different opinion on how EDA and SOA relate to each other but we are of the opinion that EDA and SOA are peers and complements, both within business and IT context. The relation between the two concepts is obvious almost in all architecture layers, from business and information to integration and technical infrastructure. It is however important to point out that despite the identified relations between EDA and SOA both concepts can be implemented separately and independent of each other. We are also of the opinion that EDA is an architectural style as SOA and we don't consider it just as an implementation style of SOA.

EDA and SOA are able to get implemented and function in parallel without any contradictions. We believe there are three main reasons for this natural collaboration. The first one is common and aligned business objectives, the second one is the nature of both architecture concepts which builds upon decoupled components and common data model, and the third reason is use of common infrastructure and technology. EDA and SOA are complementary in many aspects. By adding EDA on top of your SOA architecture new capabilities are introduced and revealed. The capabilities have been discussed and analysed in previous chapters in detail. The following table is a summary of new capabilities provided by EDA and how it cooperates with SOA.

Architecture Area	EDA extends SOA	EDA interacts with SOA
Business	Proactive business Decoupled business	EDA and SOA have common business objectives. Events and Services identified must be aligned and defined by the same business. Events and Services must be at the same granularity level.
Information	Active information Real-time information Business Intelligence Synchronized data	EDA and SOA use the same data source. EDA and SOA use the same data semantics. SOA is using EDA to distribute and publish data in real-time. EDA events are publishing new data which is

		<p>updated and managed by SOA services.</p> <p>Data synchronization can involve both EDA events and SOA services.</p> <p>Combining pushing and pulling of data when implementing BI and BAM.</p>
Information System & Integration	<p>Decoupled business services</p> <p>Sharing information instead of asking for information</p>	<p>EDA events transport “business data” and “process status” between SOA services.</p> <p>SOA services transform “business data” and “process status”.</p> <p>SOA services are decoupled by using EDA events.</p> <p>A SOA service may generate or consume EDA event.</p> <p>A SOA service must be able to identify relevant events and take immediate action.</p> <p>Events and services are both main components within integration architecture.</p> <p>EDA and SOA leverage same integration technology platform.</p>
Technology Infrastructure	Scalability	<p>Use of common technology and infrastructure capabilities e.g. ESB.</p> <p>Integrating ESB and CEP.</p>

Figure 26 Summary of how EDA and SOA interacts

Though relations between EDA and SOA are clearly listed in the table above, the challenges when implementing EDA and SOA should not be underestimated. Both SOA and EDA as described in this study are fairly new concepts and in early stages of maturity. Many companies are today struggling with implementing SOA and realizing the benefit and value promised. EDA has been around for many years but often as a technical design pattern providing scalability and high transactional performance. EDA as the business concept and architectural style described in this paper is still new, uncommon, and not supported by accepted guidelines, frameworks, and design patterns. Another major challenge is the governance of events. This is also a topic where the level of maturity is not high enough and where real-life experience is rare.

The main challenge is however to involve business when implementing EDA and SOA. In our opinion this is the key to success and will help the architect to understand where and how to use SOA and EDA.

8.2 Proposals for future research

In this study we have focused and discussed both EDA and SOA mainly at a conceptual level. We believe it would be much interesting to take this study one or two steps further, analyzing both concepts from a practical and implementation point of view.

- It would be interesting to do an empiric research, studying a number of organizations where EDA is getting implemented. The study should focus on the business goals the organization is hoping to achieve, the architecture and design patterns applied, and the technical infrastructure used.
- It is also interesting to identify organizations where both EDA and SOA are implemented or under development. The relation between SOA and EDA should be in focus within the empiric study. Evaluation of the value SOA and EDA bring to the business can also be studied and analyzed.
- Finally, to study how existing CRM and ERP systems adapt to EDA and the opportunities and challenges with it.

List of References

- Aerts, A.T.M., Goossenaerts, J.B.M., Hammer, D.K., Wortmann, J.C. (2003). *Architectures in context: on the evolution of business, application software, and ICT platform architectures*.
- Allen, P. (2006). *Service Orientation: Winning strategies and best practices*. Cambridge University Press.
- Avison, D., Jones, J., Powell, P., Wilson, D. (2004). *Using and validating the strategic alignment model*. *Journal of Strategic Information Systems* 13 (2004) 223-246
- CIO-council. *A practical guide to federal Enterprise Architecture*. (2001).
- Bieberstein, N., Bose, S., Flammante, M., Jones, K., Shah, R. (2006). *Service-Oriented Architecture Compass-Business Value, Planning, and Enterprise Roadmap*. IBM Press.
- Bracheau, J.C., Wetherbe, W.C. (1986). *Information Architecture: methods and practices*. *Information processing and management*, Volume 22, Number 6 1986, pp. 453-463
- Bryman, A, Bell, E. (2007). *Business Research Methods*, 2nd ed.,Oxford University Press
- Campbell, S., Mohun, v. (2007). *Mastering Enterprise SOA with SAP NetWeaver and mySAP ERP*. John Wiley and Sons.
- Department of the Treasury CIO Council (2000). *Treasury Enterprise Architecture Framework Version 1*. Treasury CIO Council, Department of the Treasury.
- Earl, M.J. (1989). *Management Strategies for Information Technology*. Prentice-Hall, Englewood Cliffs, New Jersey
- Erl, T. (2007). *SOA: Principles of service design*. Prentice Hall/PearsonPTR.
- Esaiasson, P., Gilljam, M., Oscarsson, H., Wängnerud, L. (2003). *Metodpraktikan*. Norstedts Juridik AB.
- Flurry, G. (2007). *Exploring the Enterprise Service Bus, Part 1: Discover how an ESB can help you meet the requirements for your SOA solution*. IBM Software Group. http://www.ibm.com/developerworks/architecture/library/ar-esbpat1/?S_TACT=105AGX04andS_CMP=ART Verified 2009-Nov-29
- Heffner, R. (2006). *EDA, SOA 2.0, and Digital Business Architecture. How Event-Driven Applications fit into the future of IT architecture*. Forrester Research Inc.
- Henderson, J., Venkatramen, N. (1989). *Strategic Alignment: A Model for Organisational Transformation*, in: Kochan, T., Unseem, M. (Eds.). OUP, New York.

- Hohpe, G., Woolf, B. (2006). *Enterprise Integration Patterns Designing, building, and deploying messaging solutions*. Addison-Wesley.
- Hoof, J.V. (2007). *SOA and EDA: using events to bridge decoupled service boundaries*. SOA Magazine Issue IV: February 2007.
- Howard, R. (2000). *The CEO as organizational architect: An interview with Xerox's Paul Allaire*. Harvard Business Review
- Hurwitz, J., Bloor, R., Baroudi, C., Kaufman, M. (2007). *Service Oriented Architecture for dummies*. John Wiley and Sons.
- Kettinger, W.J., Teng, J.T.C., Guha, S. (1992). *Business process redesign and information architecture: establishing the missing links*. College of Business Administration, University of South Carolina, Columbia
- Koshafian, S. (2007). *Service Oriented Enterprise*. Auerbach publications.
- Maes, R., Rijsenbrij, D., Truijens, O., Goedvolk, H., (2000). *Redefining Business-IT Alignment Through A Unified Framework*. Universiteit Van Amsterdam/Cap Gemini White Paper.
- Magoulas, T., Pessi, K. (1998). *Strategisk IT-management*, Göteborgs universitet.
- Mamdouh, I. (2007). *Service-Oriented Architecture and Enterprise Architecture, Part I: A framework for understanding how SOA and Enterprise Architecture work together*. <http://www.ibm.com/developerworks/webservices/library/ws-soa-enterprise1/> (verified 2009-November-29)
- Margolis, B., Sharpe, J. (2007). *SOA for the Business Developer: Concepts, BPEL, and SCA, First Edition*. MC Press.
- Marechaux, J.L. (2006). *Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus*. IBM Software Group
- Marks, E.A., Bell, M. (2006). *Service Oriented Architecture A planning and implementation guide for business and technology*. John Wiley and Sons, Inc.
- McBride, N. (1998). *Towards a Dynamic Theory of Information Systems Planning*. Proceedings of the 3rd UKAIS Conference, Lincoln University, 218-230.
- McGovern, J., Tyagi, S., Stevens, M., Matthew, S. (2003). *Java Web Services Architecture*. Morgan Kaufmann Publishers.
- Mendelson, H. (1994). *Organizational architecture and success in the Information Technology industry*. Graduate School of business, Stanford University, Stanford, California.

- Michelson, B.M. (2006). *Event-Driven Architecture Overview*. Patricia Seybold Group. <http://www.psgroup.com/detail.aspx?id=681>. Verified: 2008-May-01
- Nadler, D.A., Gerstein, M.S. (1992). *What is organizational architecture?* Harvard Business Review September–October (1992).
- Patel, R., Davidson, B. (1994). *Forskningsmetodikens grunder*. Lund: Studentlitteratur.
- Ranadivé, V. (1999). *The power of now*. Computing McGraw-Hill.
- Schulte, R.W. (2008). *Tutorial for EDA and how it relates to SOA*. Gartner. ID Number: G00155163. Publication date: 14 February 2008
- Schulte, R.W., Sholler, D. (2008). *Event-Driven SOA is driving ahead*. Gartner. ID Number: G00158985, Publication date: 26 June 2008
- Schulte, R.W., Sholler, D. (2008). *Key issues for SOA, EDA and WOA*. Gartner. ID Number: G00146543, Publication date: 26 February 2008
- Schulte, W., Gassman, B. (2009). *The role of CEP and EDA in Business Intelligence*. Gartner. ID Number: G00166324, Publication date: 13 April 2009
- Schulte, R.W., Sholler, D. (2009). *Survey Update: The value of SOA*. Gartner. ID Number: G00166447. Publication date: 24 March 2009
- Sessions, R. (2007). *Comparison of the Top Four Enterprise Architecture Methodologies*. ObjectWatch Newsletter
- Sliwa, C. (2003). *Event-driven architecture poised for wide adoption*. <http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,81133,00.html> . Verified 2009-December-12
- The Open Group (2003). TOGAF (The Open Group Architecture Framework) Version 8.1 “Enterprise Edition”. The Open Group.
- Ward, J., Peppard, J. (2002). *Strategic Planning for Information Systems*. John Wiley and Sons, LTD.
- Whittle, R., Myrick, C.B. (2005). *Enterprise Business Architecture: The formal link between strategy and results*. Auerbach publications.
- Zachman, J.A. (1996). *Enterprise Architecture: The issue of the Century*. Zachman institute for Framework Advancement (ZIFA)

Internet References

1. <http://www.ibm.com/developerworks/webservices/library/ws-soa-enterprise1/>
Verified: 2009-November-29

2. <http://www.enterprise-architecture.info>
Verified: 2009-November-29

3. http://www.ibm.com/developerworks/architecture/library/aresbpat1/?S_TACT=105AGX04andS_CMP=ART
Verified: 2009-November-29

4. <http://www.soapatterns.org>
Verified: 2009-November-29

5. <http://www.ibm.com/developerworks/webservices/library/ws-soa-eda-esb/>
Verified: 2008-May-02

6.
<http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,81133,00.html>
Verified: 2009-December-12

7. http://en.wikipedia.org/wiki/Event-driven_architecture
Verified: 2009-December-12

8. http://en.wikipedia.org/wiki/Complex_event_processing
Verified: 2008-May-24

9. <http://erpnews.ru/doc2840.html>
Verified: 2008-January-24

10. http://www.jroller.com/rameshl/entry/eda_and_cep_now_in
Verified: 2008-July-01

11. http://en.wikipedia.org/wiki/Information_technology_management
Verified: 2009-November-12

Appendix A – Zachman Enterprise Architecture Framework







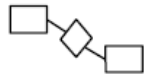
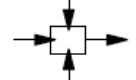
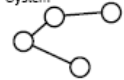



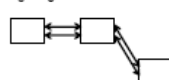
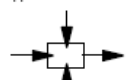
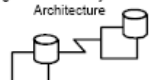


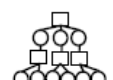
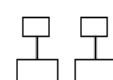

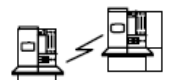

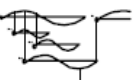







	DATA <i>What</i>	FUNCTION <i>How</i>	NETWORK <i>Where</i>	PEOPLE <i>Who</i>	TIME <i>When</i>	MOTIVATION <i>Why</i>	
OBJECTIVES/ SCOPE (CONTEXTUAL)	List of Things Important to the Business 	List of Processes the Business Performs 	List of Locations in Which the Business Operates 	List of Organizations Important to the Business 	List of Events Significant to the Business 	List of Business Goals/Strat. 	OBJECTIVES/ SCOPE (CONTEXTUAL)
<i>Planner</i>	Entity = Class of Business Thing	Function = Class of Business Process	Node = Major Business Location	People = Class of Agent	Time = Major Business Event	Ends/Means = Major Bus. Goal/Critical Success Factor	<i>Planner</i>
ENTERPRISE MODEL (CONCEPTUAL)	e.g. Semantic Model 	e.g. Business Process Model 	e.g. Business Logistics System 	e.g. Work Flow Model 	e.g. Master Schedule 	e.g. Business Plan 	ENTERPRISE MODEL (CONCEPTUAL)
<i>Owner</i>	Ent. = Business Entity Rein. = Business Relationship	Proc. = Business Process I/O = Business Resources	Node = Business Location Link = Business Linkage	People = Organization Unit Work = Work Product	Time = Business Event Cycle = Business Cycle	End = Business Objective Means = Business Strategy	<i>Owner</i>
SYSTEM MODEL (LOGICAL)	e.g. Logical Data Model 	e.g. Application Architecture 	e.g. Distributed System Architecture 	e.g. Human Interface Architecture 	e.g. Processing Structure 	e.g. Business Rule Model 	SYSTEM MODEL (LOGICAL)
<i>Designer</i>	Ent. = Data Entity Rein. = Data Relationship	Proc. = Application Function I/O = User Views	Node = I/S Function (Processor, Storage, etc.) Link = Line Characteristics	People = Role Work = Deliverable	Time = System Event Cycle = Processing Cycle	End = Structural Assertion Means = Action Assertion	<i>Designer</i>
TECHNOLOGY MODEL (PHYSICAL)	e.g. Physical Data Model 	e.g. System Design 	e.g. Technology Architecture 	e.g. Presentation Architecture 	e.g. Control Structure 	e.g. Rule Design 	TECHNOLOGY MODEL (PHYSICAL)
<i>Builder</i>	Ent. = Table/Segment, etc. Rein. = Key/Pointer, etc.	Proc. = Computer Function I/O = Data Elements/Sets	Node = Hardware/System Software Link = Line Specifications	People = User Work = Screen Format	Time = Execute Cycle = Component Cycle	End = Condition Means = Action	<i>Builder</i>
DETAILED REPRESENTATIONS (OUT-OF-CONTEXT)	e.g. Data Definition 	e.g. Program 	e.g. Network Architecture 	e.g. Security Architecture 	e.g. Timing Definition 	e.g. Rule Specification 	DETAILED REPRESENTATIONS (OUT-OF-CONTEXT)
<i>Sub-Contractor</i>	Ent. = Field Rein. = Address	Proc. = Language Stmt I/O = Control Block	Node = Addresses Link = Protocols	People = Identity Work = Job	Time = Interrupt Cycle = Machine Cycle	End = Sub-condition Means = Step	<i>Sub-Contractor</i>
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK	e.g. ORGANIZATION	e.g. SCHEDULE	e.g. STRATEGY	FUNCTIONING ENTERPRISE

Figure 27 Zachman Enterprise Architecture Framework

Appendix B – SOA implementation components

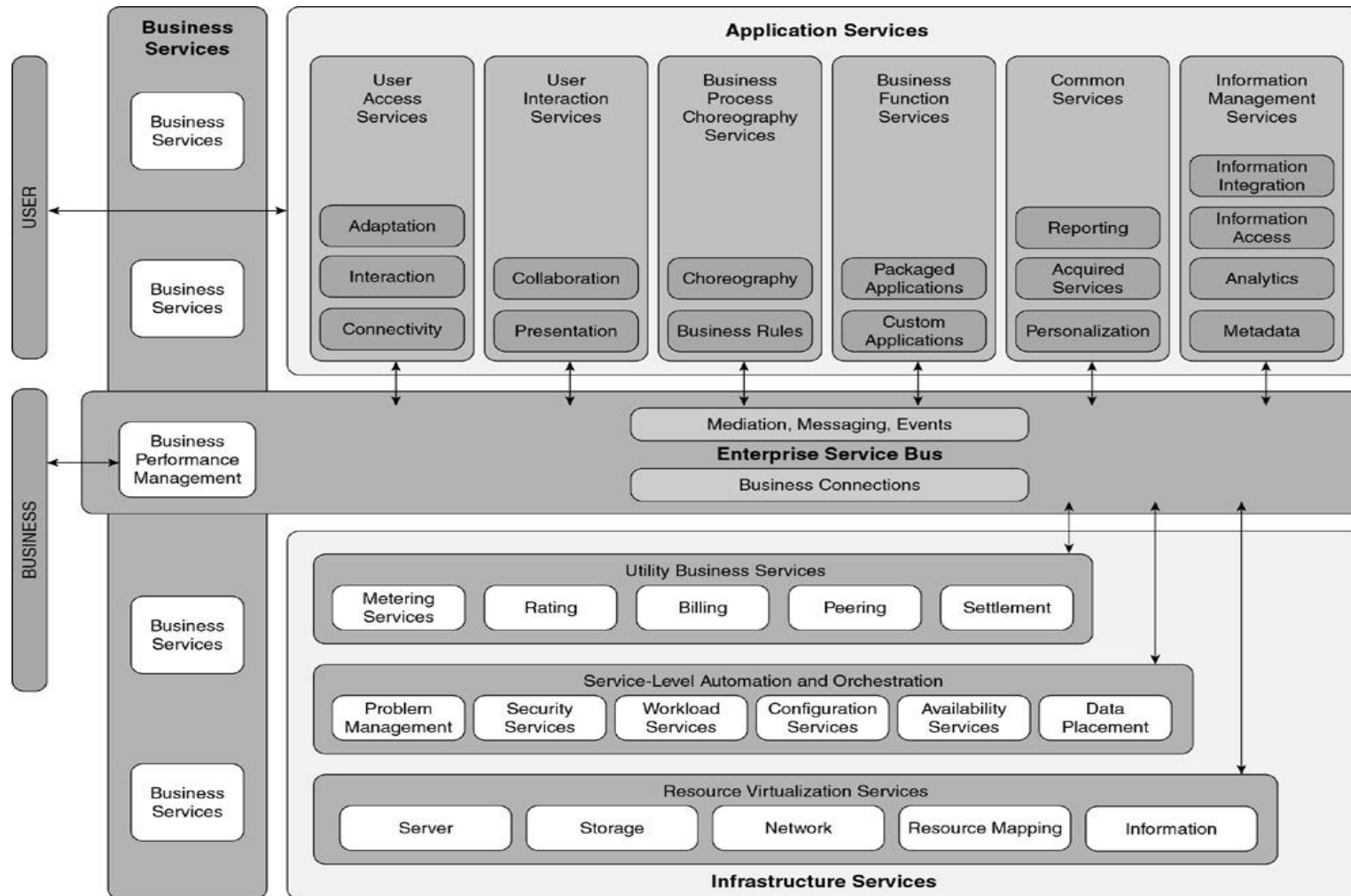


Figure 28 IBM On Demand Operation Environment (ODOE)

Appendix C – EDA implementation components

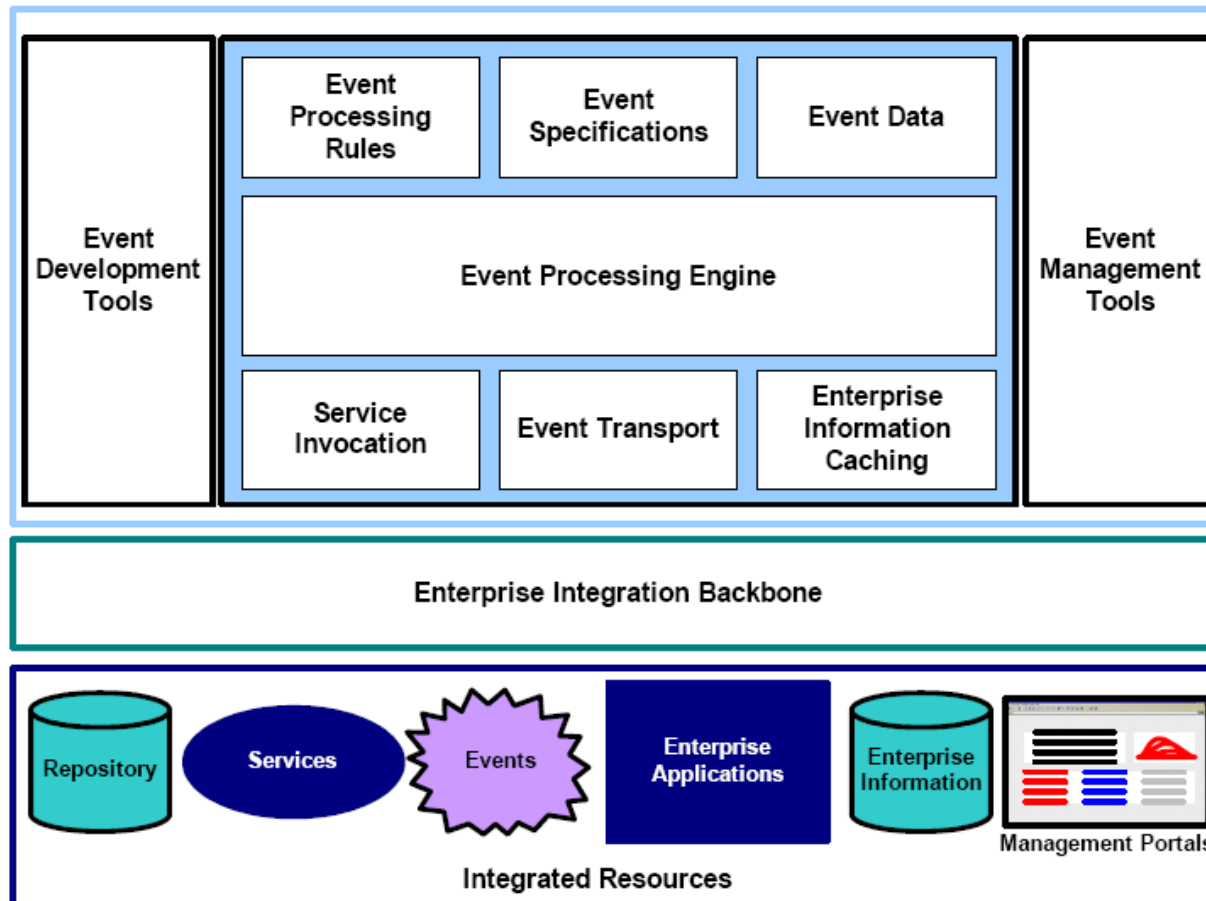


Figure 29 Major implementation components within EDA (Michelson 2006)

Appendix D – Interviews

Lennart Eriksson	2008 September 03
Senior Enterprise Architect, Onroute	

1. What are the main interaction points between EDA and SOA?

Lennart started the discussion pointing out the need of an ESB where events and services are interacting with each other. His view on this issue is that the business processes are implemented in the ESB. The process will consume or produce events which are generated based on specified business rules. The event engine/manager is implemented in the ESB platform. However according to Ericsson the challenge is defining the processes and how and when using which event.

Lennart has quite a technical view when describing EDA. From his point of view EDA is a technical approach which is much equal to asynchronous messaging implemented in an infrastructure that supports the concept e.g. ESB.

2. What are the main challenges when combining EDA and SOA?

- Governance is the major challenge when combining SOA and EDA; who is deciding when and how to change the services and events and how should you notify the involved parties? Web Service governance has been discussed as a challenge always when discussing SOA but EDA is adding an additional dimension to this problem. You should also talk about Event governance.

According to Lennart there are no standards or standard approach to handle these problems today. Some mechanism can however be custom developed to support these functions.

- Reliable messaging is another challenge. More and more sophisticated EDA will put higher requirements on messaging. No event should be lost during the process. Current messaging platforms are not reliable enough to support full EDA.
- Another challenge is how to understand the content of the events. Since producers and consumers are unaware of each other they must be able to interpret the content. Here you can either define a contract/CDM between the parties or use industry standards available.
- The last challenge mentioned by Ericsson is how to avoid an increasing and overwhelming number of events generated.

3. What are the main architecture principles when designing EDA?

Lennart described three scenarios where EDA is to consider as a good design pattern. These three scenarios could also be seen as principles when designing an EDA:

- Event producer and consumer are not aware of each other
- Great number of event producers exist which are leveraging asynchronous transactions
- Huge number of events but only a few events will be consumed by the consumers

Jan Mattsson	2008 October 07
Consultant, IT-Huset	

1. What are the main interaction points between EDA and SOA?

The main interaction point described and experienced by Jan is when SOA calls generate business events which are consumed by other modules, systems or external partners. This setup has been used as a “Gateway” to the world outside. The events generated are used to communicate with other interesting systems in the surrounding environment.

Another scenario where EDA and SOA interact is cases where events are provoking SOA calls. This is however not that common according to his experience. This may also lead to performance and/or scalability issues. The response time of the SOA calls may slow down the event flow within the EDA.

Jan describes EDA as asynchronous SOA which is loosely coupled and distributed. He has a quite technical view when discussing EDA and has mainly used EDA as a tool to reach higher degree of scalability, flexibility and modularity. According to him this is the most common way of using EDA currently in the business.

2. What are the main challenges when combining EDA and SOA?

The following challenges were discussed when implementing EDA and SOA.

- If SOA is part of EDA it may be hard to leverage the scalability benefits required
- Security is always an issue when dealing with events, e.g. who will be responsible for generating and reading events?
- Trouble management gets also much more complicated when combining EDA and SOA. A problem within the integration or the information flow process may have several reasons. Here are some of the root causes:
 - Event generator
 - Event consumer
 - Event communication infrastructure
 - Interpreting the event
- Governance is also another critical challenge dealing with services and events. Jan mentioned the following as some examples:
 - What systems or which parties are involved within an EDA?
 - How will be able to generate events?
 - What services produce or consumes events?
 - In what order and frequency should events be generated

3. What are the main architecture principles when designing EDA?

Jan describes statelessness of the events as one of the key principles when designing and building Event Driven Architecture. The main idea is to enable easy and smooth transition of events through the systems and modules. The event flow shall not be prevented or affected due to repeated status update interventions. This may heavily affect both performance and scalability of the solution.

Joshua Oyugi	2009 July 09
SOA and Integration Architect, Accenture	

1. What are the main interaction points between EDA and SOA?

Joshua has both a business and technical view when discussing EDA and starts answering this question by mentioning that SOA and EDA have a lot in common regarding business objectives and principles. SOA principles have to be applied when building EDA on top of SOA. If EDA is implemented without applying SOA principles you will not reach the high level of decoupling you may require.

Another relation is how EDA and SOA are implemented. Both can/should use the same infrastructure, technology and standards with some exceptions. Joshua is pointing out that one reason why SOA and EDA are discussed as complementary concepts is because of SOA technology and standards available which may also facilitate and support EDA implementation to some degree.

But the relation is not only limited to the technical infrastructure and standards. Also the conceptual architecture is much dependent on each other. Joshua demonstrates this relation by adding a new “event processing layer” into traditional SOA layers. The event managed through this layer can be published by services or the orchestration engine which also can consume the events published.

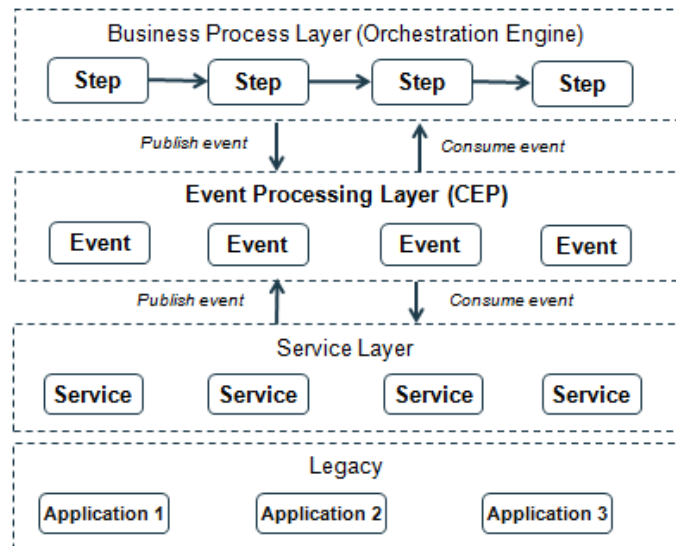


Figure 30 SOA and EDA Architecture Layers

One common enterprise data model is another area where EDA and SOA have a clear interaction and uses the same model. Joshua is however referring to the important role EDA plays when it comes to sharing real-time information. An example of how EDA extends SOA in this area is when implementing BAM (Business Activity Monitoring). Joshua is of the opinion that implementation of BAM will be much more cost effective and bring higher value if using EDA in your existing SOA environment.

Joshua is summarizing this discussion by stating that “to reach excellence in SOA – you should study EDA” and is of the opinion that SOA and EDA are complementary and that EDA will extend your SOA capabilities within the previous mentioned areas.

2. What are the main challenges when combining EDA and SOA?

Joshua is considering both business and technical challenges when answering this question. The main challenge is perhaps the event definition and bringing business into

EDA, avoiding using EDA just as a messaging pattern which is the traditional understanding of EDA.

- Event definition – The definition of what an Event is very broad which may create confusion during design and implementation of this architecture. An event has to be unique. You need to have an event dictionary to make sure your events are unique in the enterprise.
- Interoperation (cost effective way of making the EDA components interoperate) between EDA components e.g. event producer, event listener
- Defining an Event Web Service that will publish an event. Typically a Web Service is calling another Web Service but in the EDA scenario we need to create a “one-way” Web Service which is just publishing an event. The nature of this Web Service is different compared with a traditional SOA Web Service in a request/reply pattern.
- Exception handling since the publisher will not get an instant reply from the consumer
- Debugging since the architecture is decoupled

3. What are the main architecture principles when designing EDA?

According to Joshua the concept of EDA is not new but real EDA projects and implementations are very rare. EDA is not studied and analyzed as much as SOA and the principles are not summarized in a common way. Though EDA may have its own principles Joshua is referring to the importance of applying SOA principles when implementing EDA. This is of course relevant only when EDA and SOA are considered in parallel. The following EDA characteristics were mentioned by Joshua:

- Decoupling of the components – event producer, event listener, event processor, event reactor. The decoupling is done by using a messaging infrastructure e.g. an ESB
- No centralized controller (like Orchestration Engine in SOA) between event producers and consumers
- Event definition – an event has to be unique. You need to have an event dictionary to make sure your events are unique in the enterprise.
- Asynchronous messaging pattern will be used
- Statelessness of the components (the state travels with the event)
- EDA uses commonly accessible infrastructure e.g. ESB
- When implemented on an SOA, an event is a SOAP message