EKONOMIHÖGSKOLAN
Lunds universitet
Department of Informatics
Ole Römers väg 6, 223 63 Lund

Service-Oriented Architecture (SOA) quality attributes

– A research model

Master thesis, 10 credits, INF101, in Informatics
11th of October 2006

Supervisors: Hans Lundin / Erik Persson
Examiners: Magnus Wärja, Claus Persson
Author: Annika Pettersson

MANY THANKS TO…

IBM Switzerland and IBM Germany

- Gisbert Krüsemann, IBM Germany
- Norbert Furth, IBM Switzerland
- Franziska Feller Kaetterer, IBM Switzerland
- Ralf Schaarschmidt, IBM Germany

University of Lund

- Erik Persson, University of Lund
- Hans Lundin, University of Lund

UBS Switzerland

- Günther von Bülzingslöwen, UBS Switzerland

EKONOMIHÖGSKOLAN
Lunds universitet

Department of Informatics
Ole Römers väg 6, 223 63 Lund

Service-Oriented Architecture (SOA) quality attributes
– A research model

© Annika Pettersson

Master thesis presented: October 2006
Span: 98 pages
Supervisors: Hans Lundin / Erik Persson

Abstract

Background: Many of the architectural discussions and proposals, currently being held and formulated at IBM, focus on service-oriented architecture (SOA). In fact, every other day, IBM employees receive mails containing SOA information. Having in mind that IBM is one of the leading consultant and software companies in the world, as well as considering that this architecture is said to be an improvement of EAI, the interest arose in conducting a research around a potential SOA Quality Evaluation Model. With this model it should be possible to gain an indication of the extent a SOA is contributing or limiting to business benefits, in comparison to an already existing architecture. For the research area the Swiss bank UBS was chosen, as they recently implemented a service-based architecture.

Aim: To create a SOA Quality Evaluation Model that is applicable to SOA implementations.

Method: Based on secondary data, attributes that describe the quality features of SOAs were filtered out and to some extent gathered. These quality attributes were then combined with questions, to collect information about how employees, at the UBS, experience the difference between the old architecture and the newly implemented service-oriented architecture. Finally, the gathered results were weighted, presented and analyzed in the SOA Quality Evaluation Model.

Conclusion: All attributes, except 'Security', 'Efficiency', 'Reliability' and 'ROI', indicate an improvement with the replacement of the old architecture and none of the 12 quality measurements show proof of deterioration. Moreover, nine out of the 12 attributes were regarded as critical for success. Hence, in general it can be assumed that these attributes have been in accordance with what can be expected from an architecture evaluation model at a bank like UBS.

Keywords: SOA, quality attributes, quality model, UBS

TABLE OF CONTENTS

TABLE OF FIGURES

OVERVIEW OF ABREVIATIONS

AS THIS ESSAY WILL CONTAIN SEVERAL ABREVIATIONS, THIS GATHERING WAS CONDUCTED TO PROVIDE THE READER WITH A GENERAL OVERVIEW. DEFINITIONS, HOWEVER, ARE NOT GIVEN HERE, BUT IN THE RESPECTIVE CHAPTERS.

ACC	Cash Core Account
CI	Customer Information
CICS	Customer Information Control System
CIF	Customer/Central Information File
CORBA	Common Object Requestor Broker Architecture
CSF	Critical Success Factor
CSF	Common Framework Service
EAI	Enterprise Application Integration
ESB	Enterprise Service Bus
Global WM&BB	Global Wealth Management & Program Business Banking
HTTP	Hyper Text Transfer Protocol
IBM	International Business Machines Coorporation
IEEE	Institute of Electrical and Electronic Engineers
IP	Internet Protocol
ISO/IEC	International Standardization Organization / International Electronical Commission
IT	Information Technology
J2EE	Java 2 Platform Enterprise Edition
JMS	Java Message Service
MAP	Multi-Channel Access Platform
MQ Series	Message Queuing Series
MSMQ	Microsoft Message Queuing
OASIS	Organization for the Advancement of Structured Information Standards
OLU	Open Lan Unix
OO	Object-oriented
OSI	Open Systems Interconnection
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
ROI	Return on Investment
RPC	Remote Procedure Call
SMTP	Simple Mail Transport Protocol
SOA	Service-oriented Architecture
SOAP	(former: Simple Object Access Protocol)
SSP	Strategic Solution Program
UBS	(former: United Bank of Switzerland)
UDDI	Universal Description, Discovery and Integration
W3C	World Wide Web Consortium
WAS	WebSphere Application Server
WebSphere MQ	WebSphere Message Queuing
WMA	Cash Dispenser
WS	Web Service
WSDL	Web Service Description Language
XML	eXtensible Markup Language

INTRODUCTION

TO GIVE THE READER A GOOD START INTO THE TOPIC, THE INTRODUCTION WILL FOCUS ON THE CURRENT IT SITUATION AND THE REASON FOR ENTERING A NEW ERA. THE ADDITIONAL ENLIGHTENING OF THESIS DETAILS, SUCH AS PREFACE, AIM, DELIMITATION AND DISPOSITION, ARE TO GET A MORE SPECIFIC UNDERSTANDING OF THE THESIS CONTENT AND STRUCTURE.

BACKGROUND

Today, IT architectural discussions no longer evolve around Monolithic Architectures (pre 1950s up until 1960s), Remote Procedure Calls (RPCs) (1970s until the middle of the 1980s), Remote Object Invocation (the 1980s until the middle of the 1990s) or Message Processing (mid 1990s until the early 2000s). Neither is the so called Enterprise Application Integration (EAI) of same interest anymore as in the late 1990s. In fact, it seems as if the IT-world is once again embracing a new architectural approach, i.e. Service-Oriented Architectures (SOA). (*Bergmann, 2005*)

Due to the technical evolution it is to some extent, relatively self-evident that SOAs can be used to implement EAIs. However, that EAIs with technical features, such as messaging, message brokers and message busses, can be used to implement SOAs might seem a bit more surprising. In fact, this challenges the whole reason, for introducing and evolving this new service-oriented architecture. Hence, what exactly is the main reason for this architectural evolution, besides competing on the architectural market and gaining or increasing possible market shares?

EAI did solve the severe problem of coordination between business requirements and technology, but then again it failed in addressing the complex array of integration issues. Moreover EAI could not deliver the needed business flexibility and the presence of an understandable language between business and IT. (*Bieberstein et. al., 2006*) Considering these shortcomings, it might not be too surprising that one exclaimed a 70% failure of all EAI installations during an EAI symposium in 2003. As it seems, EAI projects went over budget, missed deadlines and failed within service- and quality delivery. Furthermore the EAI implementations were said to be too complex and difficult. (*Weisser, 2004*)

The problem of missed deadlines and overrun budgets is a well-known IT problem. If one considers the results from the Standish Group in the year 1994, only 16% of all software projects (both developed from scratch and standardized applications and components) in the U.S. were successful. Another 53% of the projects were completed, but over the time estimated and budget given. The final 31% of all projects were cancelled at some point. Even though these figures had improved to some extent up until the third quarter of 2004, still only 29% projects were successfully accomplished, while 53% were still challenged and 18% not accomplished at all. (*The Standish Group International Inc, 2004*) Considering that EAIs were introduced in the mid 1990s one would have expected a significant reduction in the overall failure of software projects. Instead the reduction only reached an amount of 13% (84%-71%). Furthermore, having the major drawbacks of unavailable business flexibility and

commonly used language between technical- and business oriented work forces a new architectural approach simply had to enter the market.

Lamont (2006) claims that SOA "[…] allows for more flexible, rapid and inexpensive incorporation of new functions […]" through "[…] standardized software interface(s) to which many applications can connect." (*Lamont, 2006, p. 20*). Con-way Transportation Services, as a real life company with an implemented SOA, appreciates this close interaction between IT and business, which enables them to execute its business processes on a very coarse grained level and thus improving the business and product flexibility. Furthermore, Con-war mentions the stable architecture and maintained business logic, despite newly added business logics due to the changing market, during the eight years of implementation. (*Gruman, 2006*) In other words, a SOA is to provide the businesses with a "meet-in-the-middle" technique for business and IT.

International Business Machines Cooperation (IBM) is an IT company that has made great contributions to the development of SOA. Besides creating managing, construction and modeling applications like e.g. Customer Information Control System (CICS) and different kinds of WebSphere and Tivoli, IBM also has focused on the infrastructure, such as the SOA Cookbook and/or SOA Compass. (*Reinitz, 2003, 2004, 2005*)

PREFACE

During the six months of internship at IBM, the opportunity of focusing on a very hot topic within systems engineering was given. More precisely, the opportunity involved an extended research within the still relatively undiscovered area of SOA. Out of this perspective, it was regarded as highly interesting to focus on conducting a quality model that can be used in describing the difference between an old existing architecture and a newly implemented SOA out of a quality perspective.

Even though this approach will not specifically explain to what extent SOA is an improvement in comparison to EAI, as discussed in section *Background*, it will give an indication of the extent a SOA is contributing or limiting in comparison to an already existing architecture. Hence, if the old architecture happens to be an EAI, surely also the comparison will result in the interesting answer evolving around EAI and SOA. As such a research could not be conducted, UBS as an IBM customer was chosen. UBS was regarded as being interesting as they recently implemented a service-based architecture. Hence, at UBS it will be possible to apply a model asking for the quality differences between an old and a new architecture.

AIM AND RESEARCH QUESTIONS

Expected success from an implementation is usually put in relation to specific and measurable attributes, the so called quality attributes. By doing so, one can compare the architectural state of the old system with the new one. To be able to present this comprehensively and clearly laid out, a model will give the best preconditions. Hence, the aim of this thesis is:

To create a SOA Quality Evaluation Model that is applicable to SOA implementations.

To support the approach of reaching the formulated aim, the following research questions have been put forward:

- **What impact has the newly implemented service-based architecture had on UBS?**
- **What quality attributes can be regarded as SOA quality attributes?**
- **Is the SOA Quality Evaluation Model applicable to SOA implementations?**
- **Does the interaction between the quality attributes affect the overall outcome of the model?**

DELIMITATION

As partly already mentioned in the *Preface* this thesis will not cover specific functionalities and features of SOAs. The term SOA per se will be described on a general basis and literature will be gone through to gain the needed SOA knowledge, but the primary focus will be on creating a quality model with SOA attributes.

Moreover, the model will be applied at UBS and the conclusions drawn will be on behalf of the quality results being generated through the overall model, as well as each separate attribute. A detailed evaluation of the service-oriented impact on UBS's business, however, will not be provided, as this would require a much deeper research within UBS.

DISPOSITION

The disposition being chosen in thesis is based on the classical research approach, i.e. with an introduction, a theoretical foundation, an approach description, an empirical presentation with applied analysis and finally the overall conclusion. In addition, as this quality model is very closely intertwined with research the company, the need of providing background information about the business, the old architecture and the newly implemented one seemed essential. Due to these circumstances an additional chapter, called *Research area – UBS*, was added.

Introduction:	This chapter introduces the short historical background, current status of important architectures, as well as the overall subject.
Methodological approach:	This chapter shows how the model is created and then tested by the selected respondents within UBS.
Theoretical framework:	This chapter initiates the constantly returning concepts SOA and quality attributes.
Research area – UBS:	This chapter introduces UBS as a business and its architecture.
Results and analysis:	This chapter has its main focus on presenting and evaluating the results being gained through the model at UBS. Moreover, the model per se is discussed.
Conclusion:	This final chapter presents a short summary of the *Results and Analysis* chapter, as well concluding words about the research being conducted. In addition, also future research propositions are presented.

METHODOLOGICAL APPROACH

THIS CHAPTER PROVIDES THE READER WITH DETAILS ABOUT THE APPROACHES USED AND TO WHAT EXTENT THESE APPROACHES REALLY HAVE CONTRIBUTED TO ACHIEVE THE AIM OF CREATING A SOA QULAITY EVALUATION MODEL.

Having the aim of creating a SOA Quality Evaluation Model demands a suitable research area, strategy and the actual research method. To achieve this, this section will start with the description of the selected research area and continue with introducing the research questions, as well as the overall strategy. Having presented these aspects, the remaining parts of the chapter will evolve around the method that was chosen, i.e. how quality attributes that describe the quality features of SOAs were selected and how the questionnaire was created and applied. Finally, the gathered results were weighted and presented in the SOA Quality Evaluation Model.

SELECTION AND RESTRICTIONS

As Kvale (1997) states, both the kind and range of research area plays a great part in a research. The area being used within this research was selected as a consequence of the following statement:

- Switzerland is a representative country, when it comes to banks (see chapter *Research Area – UBS*)
- Currently, UBS is the only bank in Switzerland that has implemented a SOA in the complete company. *(Furth, 2006)*
- At the bank Credit Suisse all the mainframe applications are packed with services, which solemnly are provided outwards. Hence, within the mainframe no services are provided and therefore limits the correlation to a SOA. *(Furth, 2006)*
- UBS is a customer of IBM, supporting this research.

The sampling range of a research might either be too wide or narrow and therefore affects the ability of generalizing the results being gathered. Considering that quantitative studies usually aim at statistical significance, the classical sampling ranges within quantitative studies usually exceed the amount used in qualitative studies. *(Kvale, 1997; Miles & Huberman, 1997*) Despite this, the research area will be based on one bank only, namely UBS. Furthermore, the respondents being chosen, due to the role within UBS, were kept relatively restricted. As it was regarded that a large number of respondents at UBS would not reveal more than a small number, it was regarded adequate to receive answers from five employees at UBS. The aim was to get in contact with respondents, having both technical and business related occupations. At the UBS, these turned out to be so-called Business Architects, Application Architects or ICT Consultants

Respondent	Company	Occupation
1-5	UBS	Business Architect/ Application Architect/ ICT Consultant

Figure 1: Overview of respondents.

The only restriction with the UBS choice turned out to be the currently not yet implemented Front-End. Hence, the Front-End quality attribute usability will not be able to be specified as the other quality attributes.

RESEARCH QUESTIONS

To be able to achieve the aim set up for this thesis Miles & Huberman (1994) suggest the setting up of supporting research questions, that generally speaking are said to "[…] represent the facets of an empirical domain […]" *(Miles & Huberman, 1994, p. 23)*. Moreover, these questions are to support the overall research in terms of providing guidelines.

The connection between the overall aim and research questions, as well as the incentives for the chosen questions is presented in the following figure:

Aim	Research questions	Incentives
To create a SOA Quality Evaluation Model that is applicable to SOA implementations.	What quality attributes can be regarded as SOA quality attributes?	Being able to define quality attributes for the research model, already existing attributes, as well as SOA related features, have to be considered and evaluated.
	Does the interaction between the quality attributes affect the overall outcome of the model?	Choosing specific attributes and putting these into a model, generates a certain correlation of the attributes. This correlation is regarded as an essential part of the overall model.
	What impact has the newly implemented service-based architecture had on UBS?	As the model provides the user with results showing differences between two architectures, an interesting point to be addressed is the drawbacks and/or advantages of the new architecture.
	Is the SOA Quality Evaluation Model applicable to SOA implementations?	Aiming at creating a model being applicable to SOA implementations requires the some kind of confirmation.

Figure 2: The motivation of applying the elected research questions.

STRATEGY

As can be deduced from the research questions, the research on a general basis can be described as a so called **survey research.** In other words, the intentions are not to fulfill an experimental research, but rather a **standardized information gathering,** i.e. the respondents will receive the same structured questions under similar conditions. (*Lundahl & Skärvad, 1982, 1999*)

As survey researches are a specific part of **quantitative studies**, the approach used differs somewhat from **qualitative studies**. The main difference lies in the time spent on the different development phases and the formulation of the chosen method. For quantitative studies this means that the greatest amount of time spent during the survey research will be within the preparation and formulation phase. Furthermore, also the formulation of quantitative questions differs from qualitative questions. This mainly in terms of less focus on the perception and emotional experience of the respondents and instead more emphasis on standardized structures that generate a specific result. The distance between the researcher and the respondent is considered to be close (insider) in qualitative studies and distant (outsider) in quantitative studies. A final and important difference occurs in the relationship between theory and research. Quantitative researches are to acknowledge or confirm, while qualitative studies rather generate a result that evolves during the research. *(Bryman, 1997; Kvale, 1997; Miles & Huberman, 1994)*

CHOICE OF METHOD

A case study approach is mainly **descriptive**, i.e. the aim is to gather data that reveals the situation of the phenomenon, or more precisely, of the architecture. Hence, the approach has to generate answers about the current and old situations for the researched area, as well as pointing out to what extent a specific quality attribute is of importance for the company. As the aim in this research was to find a method supporting the ability of describing, explaining, discovering and being experimental in the range of SOA quality attributes, the **questionnaire approach** within case studies was regarded as the most appropriate one. Other case study approach such as personal interviews, written material and/or observations demand direct communication, presence or limited interaction with a research area. *(Kvale, 1997; Miles & Huberman, 1994)*

As some kind of interaction with the research area had to be conducted, only focusing on written material was out of the scope. Moreover, observations would be too subjective and not using the knowledge that respondents within a specific area usually posses. Finally, the personal interviews were excluded, as these do not provide the opportunity of asking complex and long questions with similar answers. Surely also interview questions can be formulated to generate specific standard answers, but the fact still remains that questionnaires have more of a closed and structured touch. Also, the interview atmosphere has a tendency to affect the overall answers of the respondents. The drawback with questionnaires, on the other hand, is the fact that it is not possible to verify that the respondent has not been cooperating with other respondents during the questionnaire completion. *(Miles & Huberman, 1994)*

Assuming that the questions are structured in the questionnaire, the answers being generated automatically follow a certain pattern and hence, anybody could hand out the questionnaire and also analyze the results being gathered. Hence, a questionnaire does not demand the

presence of a data collector and is therefore also suitable for being handed out to several respondents. *(Kvale, 1997)*

In summary, the method chosen ended up in providing a **questionnaire and a standardized model**.

THE APPROACH

Having defined the strategy to be followed and the method to be applied, the overall preparations for the research, to be conducted, had to be made. In other words, attributes had to be selected, model structures had to be evaluated and a questionnaire had to be created.

GATHERING SOA QUALITY ATTRIBUTES

As the most essential contributor to the SOA Quality Evaluation Model, a great amount of time was invested in the research of quality attributes within the areas of:

1. **General and already partly standardized software quality attributes**
   - from McCall (1977), Boehm (1978), Grady & Caswell *(1987)*, ISO/IEC 9126 (1994), Bass et.al. (1998), Bencher (1994) and Kan (2003)

2. **Attributes, representing SOA**
   - from Roik & Balzer (2004), Bieberstein et.al. (2005; 2006), Reinitz (2003, 2004, 2005), IBM Software University (2006), Blakely (2002), Arsanjani (SOA Centre of Excellence) (2006), Keen et.al. (2004), Lager (2006), Gruman (2006), Schulte (2002) and Natis, Y & Schulte, Roy, 2003

3. **Concepts, being especially of interest for the UBS before and during the introduction of the new system**
   - Ebner (2003), UBS Business & Application Architecture (translated by Furth, Norbert) (2004), Escher (2005), Architecture & SSP (May 2005; July 2005) and Business & Application Architecture (2004)

4. **Attributes being mentioned in business articles**
   - Tuner, K. (2003), Grey et.al. (2003), Schmelzer (2005), Langel (2004), Wolfe (2003-2005) Young & Biz/ed (1996-2006) and Blakely (2002)

During the complete research a matrix was used, representing all the attributes being found and regarded as essential. Even though several attributes were divided up into sub-attributes by the authors, companies or centers, this was unaccounted for in the matrix. This approach was chosen to minimize the focusing on hierarchal structures when choosing the final quality attributes. The only grouping that was used was in terms of technical and business perspectives.

The reason for choosing the technical and business oriented perspectives in the model, has its foundation in the way a SOA is said to act, i.e. the discussion about being an architecture combining both IT and business (see section *Service-oriented architecture (SOA)*).

To be able to select the most suitable attributes in the end, each attribute was marked with a value of one. Besides that, a segmentation, where the just mentioned area 1. was separated

from the areas 2, 3 and 4, was conducted. This separation was done to provide a better overview what is considered important by UBS, business and SOA articles. In the end each segment was summarized, ending up in an overall total for each attribute.

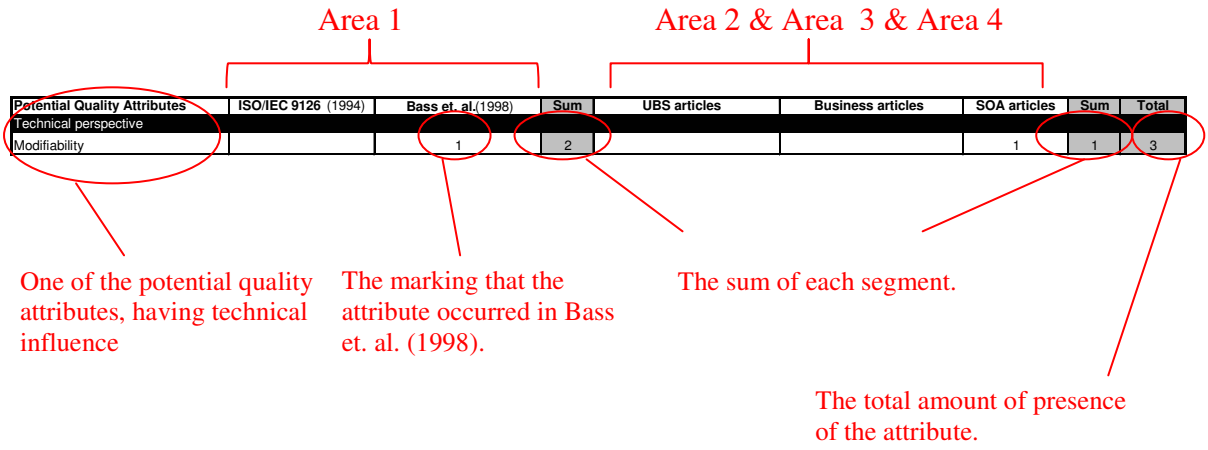


Figure 3: An extract, from the *Attachment 1: Overview Matrix of Potential Quality Attributes*, with explanations.

Having marked all possible attributes and pointed out how frequent these attributes turned up during the bibliographical research, the next step was to evaluate which of theses attributes that were to be entered into the SOA Quality Evaluation Model. This evaluation was based on the frequency of appearance in the read texts, as well as personal judgment.

Surely, this approach can be challenged, as the secondary data was limited and not specified by any scientific resource. On the other hand, the used references, such as IEEE, IBM and UBS, are well known experts within their respective area and should therefore posses the acknowledgement needed in this context. The fact that not the same amount of secondary data, such as papers and/or books, could be used for each expert reference simply has to be referred back to the fact that the experts do not have the same preconditions when it comes to published material. Despite these preconditions, the chosen secondary data was selected with the aim of minimizing any potential expert emphasis.

The fact that ‘Area 1’ mentions all its references and ‘Area 2 & Area 3 & Area 4’ are combined should not be a matter to challenge in terms of weighting, as the latter mentioned areas regard the presence of the quality attribute by presenting a greater value in the concerned cells.

To what extent it is justifiable to use a personal judgment can always be challenged, but considering that not all attributes were predefined, a subjective, but well informed, declaration was regarded as satisfactory approach in this evaluation and selection step.

EVALUATING SOA QUALITY ATTRIBUTES

Having gathered all potential attributes in the matrix the next step was to choose the ones, being suitable for the SOA Quality Evaluation Model

The selection was based on the following:

1. **Personal interpretation**
2. The **total amount of presence** in both segments.
3. The **presence in the areas 2, 3 and 4**

The personal interpretation was used to, in combination with bundling suitable quality attributes. In the bundling process, spread attributes described similarly, were gathered. Examples are, the following:

Bundled Potential Quality Attribute	Potential Quality Attributes
Modifiability	modifiability, maintainability, changeability
Functionality	functionality, capability
Security	security, integrity
Integrability	integrability, installability, interoperability
Efficiency	efficiency, business efficiency, performance, time to market
Reliability	reliability, availability, recoverability
Portability	portability, replaceability
Reusability	reusability, asset reuse
Return on Investment (ROI)	return on investment (ROI), time to market, revenue
Flexibility	adaptability, competition, business agility
Development costs	development cots, development time, project costs, project time, integration expenses
Usability	usability, learnability, understandability

Figure 4: The quality attributes being bundled.

After this bundling the names of the bundled potential quality attributes were considered. All attribute, but flexibility, seemed reasonable and hence maintained the same name. For flexibility, on the other hand, the name 'Business flexibility' was considered as more correct. The bundled potential quality attributes, with updated names, are all present in *Attachment 2: Bundled overview matrix of potential quality attributes*. As can be noticed the marking has still been maintained, i.e. also where the bundling has been taking place the marking has simply been summarized within the concerned cell.

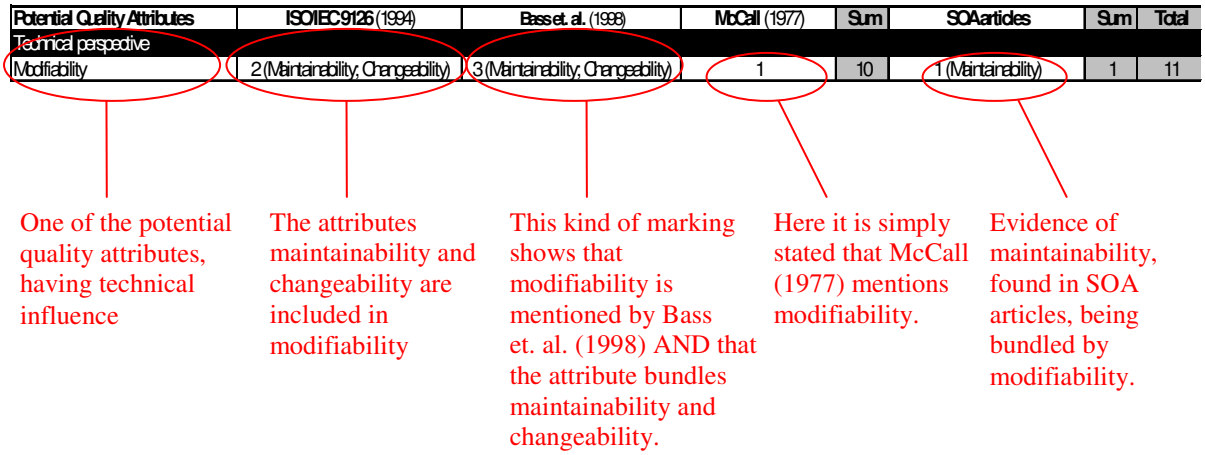


Figure 5: An extract, from the ***Attachment 2: Bundled overview matrix of potential quality attributes***, where it is described how the bundling is to be understood.

Having done the bundling the next step was to focus on the total amount of presence of each and every potential quality attribute. This approach was easily done, due to the already marked attributes. In summary, all attributes having the total sum of more than 5, were directly added into the SOA Quality Evaluation Model.

Quality Attributes	Total
Modifiability	11
Integrability	7
Efficiency	9
Reliability	10
Portability	6
Reusability	7
Return on Investment (ROI)	9
Business flexibility	6
Development costs	9
Usability	9

Figure 6: Potential quality attributes that were regarded as real quality attributes for the SOA Quality Evaluation Model, due to the total amount of presence in the bibliographical research.
(see ***Attachment 3: Overview matrix of selected quality attributes***)

Other potential attributes being regarded as essential for the SOA Quality Evaluation Model, were defined through the presence in the columns presenting SOA, UBS and business articles. Having more than one mark was considered as being essential for the overall model. After all, one mark does not only indicate that only one UBS, business or SOA article mentioned the attribute. As presented in chapter *Gathering SOA quality attributes*, all three areas are made up of several literature sources, which also shows that the attribute might have occurred more than once during the bibliographical research. The reason why these three areas were not treated as the first segment of the matrix, was simply due to the fact that the desired overview would have gone lost by presenting each and every one of the articles. This, in turn, also explains why these three columns weigh a bit more in this final selection stage.

Additional attributes being added to the SOA Quality Evaluation Model are thus security and scalability:

Quality Attributes	UBS articles	Business articles	SOA articles	Sum	Total
Security	1		1	2	4
Scalability	1		1	2	4

Figure 7: Extract from *Attachment 3: Overview matrix of selected quality attributes*, showing that the attributes security and scalability are added due to the presence in the column UBS and SOA articles.

In summary the following 12 attributes were selected to be entered into the SOA Quality Evaluation Model:

Quality Attributes
Modifiability
Security
Integrability
Efficiency
Reliability
Portability
Reusability
Scalability
Return in Investment (ROI)
Business flexibility
Development cost
Usability

Figure 8: The selected quality attributes.

MODEL STRUCTURE OF THE SOA QUALITY EVALUATION MODEL

Presenting the gathered results can be visualized in different kinds of ways. One can, for example, use matrices, diagrams, figures and/or models. The literature on the subject covered, for example, product quality models such as McCall *(McCall et. al., 1977)*, Boehm *(Boehm et al., 1978)*, FURPS *(Grady & Caswell, 1987)* and ISO/IEC 9126 *(ISO/IEC, 1994)*. A somewhat different approach to the just mentioned, is the one by Arsanjani and the SOA Centre of Excellence (2006). The so called Service Integration Model, by Arsanjani and the SOA Centre of Excellence (2006), uses a kind of Component Business Modeling (CBM)-map to provide the user with an overview of the "[…] current state in service integration and flexibility (including services orientation) and their desired or future state, for a line of business or enterprise." (*Arsanjani, 2006, p. 4*) For this stars and arrows in between the stars are sued to show where the starting point and the point to be achieved is situated:

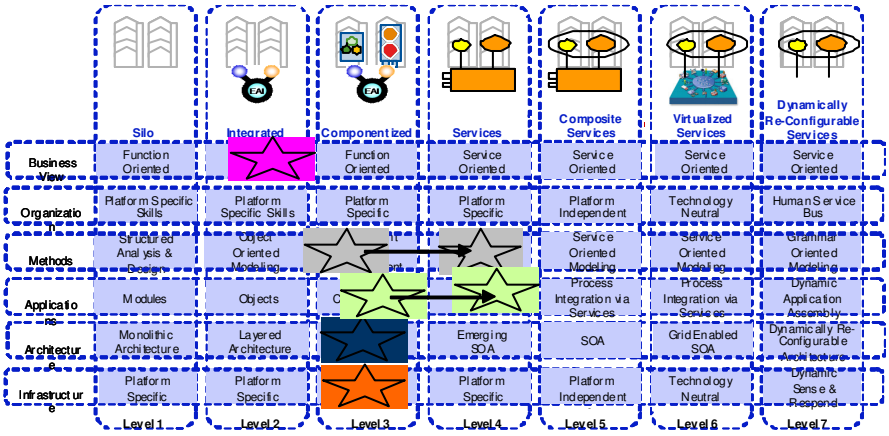


Figure 9: SIMM as an alternative of presenting gathered results. *(Arsanjani, 2006, p. 7)*

Surely this could have been one way of presenting the material, but since the SOA Quality Evaluation Model is combined with questions that generate a value between zero and four (see section *Questionnaire for the SOA Quality Evaluation Model*), the model has to be more applicable to the circumstances. A technique considering this is the so called Spider web technique (Inspired by: *Krüsemann, 2006*).

In this model each spanning line represent an attribute at each end, while the interconnecting lines represent the answer being giving by the respondents. Furthermore, the model shows a scale of zero to four, where two is neutral, i.e. all answers being less than two indicate a deficit in the architecture and all answers exceeding two are a positive feedback.

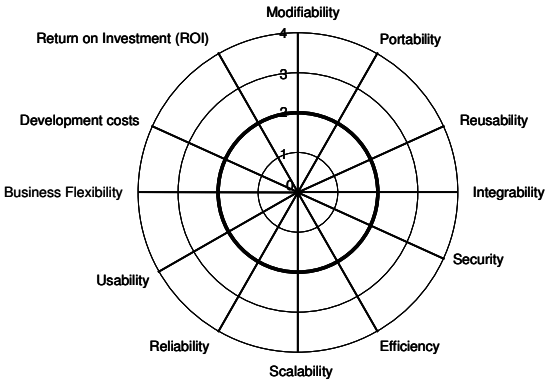


Figure 10: An example of a Spider web.

The intention with this approach is to be able to add answers gained for both the old architecture and new one. Even though the model will be able to present all gathered results at once, the choice was made to go through the following steps, before reaching the overall Spider web, alias SOA Quality Evaluation Model:

1. Create a model with the **gathered results of the old architecture**, i.e. having gathered and added all answers from the respondents. These results were represented with a red line.

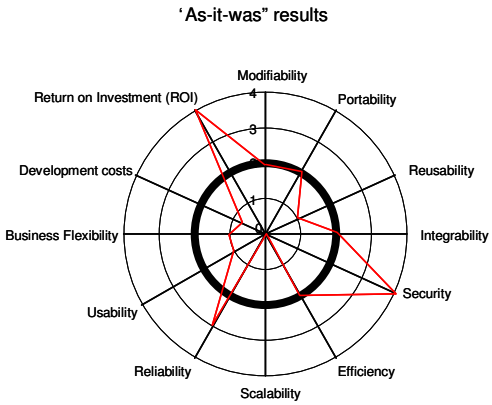


Figure 11: An example of a Spider web with gathered results about an old architecture.

2. Create a model with the **gathered results of the new architecture**, i.e. having gathered and added all answers from the respondents. These results were represented with a green line.

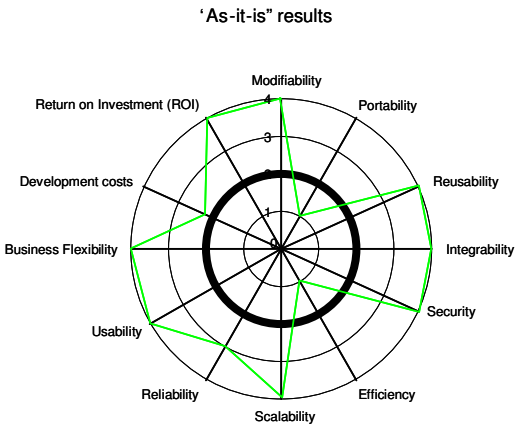


Figure 12: An example of a Spider web with gathered results about a new architecture.

3. Create a model with the **gathered results of the importance of the attributes**. These results were represented blue dots along the lines.

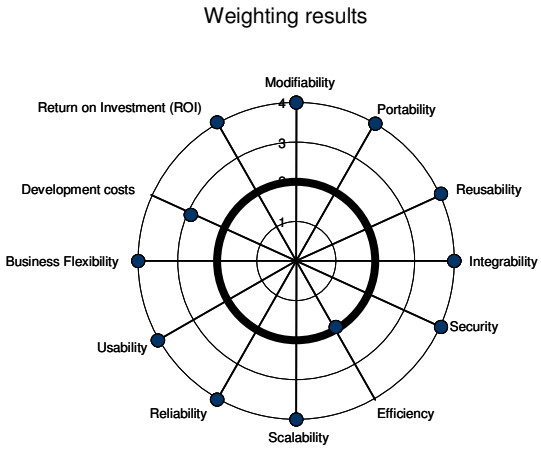


Figure 13: An example of a Spider web with gathered results of how important a company considers the attributes to be.

4. Create a model **including all aspects from the points above**.

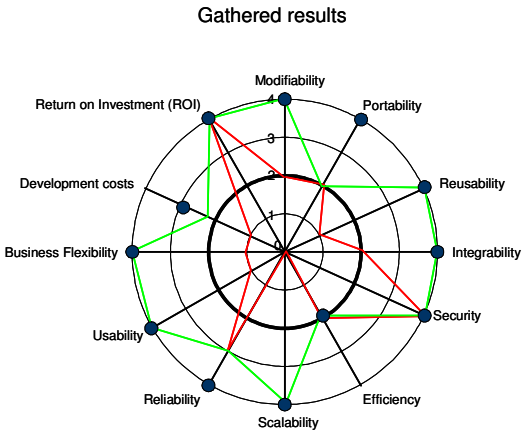


Figure 14: An example of a Spider web presenting the results from an old and a new architecture and importance of the different quality attributes.

In summary, this Spider web approach shows the difference of quality between the two architectures, as well as to what extent a specific attribute is of importance for the success of the company.

QUESTIONNAIRE FOR THE SOA QUALITY EVALUATION MODEL

As the aim was to create a descriptive questionnaire, where it would be possible to gather a wide range of information about the old and new architecture, as well as the critical success factors (CSFs) (see chapter *Theoretical framework*) of the company, the questionnaire was created structurally. In other words, both **questions and answers were structured** and formulated in advance, so that the questionnaire did not allow any specific interpretations and freely formulated answers. Furthermore, the quantifying of answers was specified before the questionnaire was handed out. *(Bryman, 1997)*

Every attribute was considered covered through **two questions**, where the first one focused on the importance of the attribute and the second one on the old and new architecture. Surely, one could have formulated two questions to discuss the old and new architecture, but the fact remains that these two questions would have been as good as identical in formulation. Hence, the risk that the respondents would tend to ignore specific details and get bored was relatively high. The first questions were defined as Weighting Questions and the other once as 'As-it-was' and 'As-it-is' Questions:

Questions
a) Weighting Question
b+c) 'As-it-was' and 'As-it-is' Question

Figure 15: An extract from *Attachment 5: Questionnaire – SOA Quality Attributes* presenting the two question alternatives.

For the answers, an approach was chosen were not only five different answer alternatives were given, but each answer was additionally defined. In other words, the respondent should be fully aware of the choice he/she is making. All answers were divided into a scale, typically used in questionnaire, i.e. the **Likert scale** (*Kvale, 1997*). Thus, both question alternatives could be answered with either:

- 'Yes, I strongly agree'
- 'Yes, I agree'
- 'I neither agree nor disagree'
- 'No, I disagree'
- 'No, I strongly disagree'

Likert scale, Answer alternatives				
Yes, I strongly agree (4)	Yes, I agree (3)	I neither agree nor disagree (2)	No, I disagree (1)	No, I strongly disagree (0)

Figure 16: An extract from *Attachment 5: Questionnaire – SOA Quality Attributes* presenting the five answer alternatives.

By having a five step scale, the respondent has the opportunity to choose between affirming and two denying alternatives. Even though the questionnaire was structured and therefore to some extent compellent, it was still considered important that the respondent could chose a

neutral answer and thus showing that he/she does not have knowledge about the question being asked or that the question is not really relevant for the company. If one would have taken away this answer alternative the respondent would have been forced to answer something that might not be fully correct.

The explanations for each Likert scale answer are divided into two kinds, i.e. both weighting and architecture related, just like the questions. By doing so the same scale can be used for both questions, even though they in fact should have required a separate scaling:

Yes, I strongly agree (4)	Yes, I agree (3)	I neither agree nor disagree (2)	No, I disagree (1)	No, I strongly disagree (0)
Modifiability is vital for the business. (Modifiability is a critical success factor (CSF).	Modifiability is important, but not vital.	The company does not make any statement about this attribute.	The company does not consider modifiability as being important, i.e. modifications are to lead to new architectural structures.	Modifiability is of no interest whatsoever for the company.
The architecture supports business agility and has a open structure.	The architecture is relatively flexible and allows most modifications.	The question cannot be answered.	The architecture does not support changes, without affecting the architectural structure.	The architecture has a closed and/or extremely complex architectural structure and thus cannot support any modifications.

Figure 17: An extract from *Attachment 5: Questionnaire – SOA Quality Attributes* presenting the five answer alternatives and the explanations for the quality attribute modifiability.

The order of the questions is usually of great importance as well. Lundahl & Skärvad (1982, 1999), for example, suggest that one is to begin with easy and pleasant questions, before the more difficult and perhaps even unpleasant questions are asked. As the quality attributes are not regarded as being able to make the respondent feel uncomfortable, the order does not have any specific meaning. The only real order present is in terms of technical and business related questions, i.e. all technical questions are sub sequentially asked and then the same principles applied on the business questions.

THE "HAND-OUT"

After having received and considered feedback, concerning the SOA Quality Evaluation Model, from both IBM employees and supervisors at the University of Lund, the final step was to hand out the questionnaire to the respondents.

The hand-out was planned to be conducted via mail, after having received the names of the respondents. Due to changed circumstances, however, the hand-out was not performed directly to the respondents. Instead a contact at UBS received the instructions (see *Attachment 4: Introduction for the respondent*) and questionnaire and distributed these to the respondents selected during a session. During this session each respondent considered answers of their own. These answers were then gathered and returned in a consolidated form, due to UBS request.

APPROACH OF THE RESULTS AND ANALYSIS

Being able to present and analyze the research results being gained, Miles & Huberman (1997) suggest a specific data management approach, with the outcome of being able to present the gathered results in an easy, structured, reliable and flexible manner. Having this in mind and moreover being aware of the fact that quantitative data results already have a relatively easy and comprehensive structure, the data was gathered and placed in the overall Spider web being presented in section *Model structure* and *Attachment 7: the SOA quality*

evaluation model. In addition, figures with the explicit results, i.e. the Likert scale results, were provided for each attribute:

a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
4	2	4

Figure 18: One of the figures presenting UBS results for the analysis.

For the analysis the foundation chosen was on the one hand based upon the Spider web and the Likert scale results, but on the other hand also on the research questions, being discussed in section *Aim and research questions*. The research questions can be regarded as both a kind of code of practice throughout the whole analysis, as well as separate discussion part of the analysis. The latter mentioned aspect is especially applicable to three of the questions, where also separate sections were formed.

In the sections covering the extent of the attributes being SOA quality attributes (see *Technical perspective* and *Business perspective)*, a further figure principle is used to ease the understanding of what attributes are obvious, potential or less appropriate in the context. This figure is based on the Likert scale approach, for weighting and 'As-it-is' questions, being mentioned in section *Questionnaire for the SOA Quality Evaluation Model* and is simply combined with the three descriptions of applicability:

Likert scale value:	4	3	=< 2
Quality and SOA applicability:	Obvious	Potential	Less appropriate

Figure 19: The approach used for dividing all attributes in being obvious, potential or less appropriate SOA quality attributes.

VALIDITY AND RELIABILITY

As the SOA Quality Evaluation Model, representing both the selected SOA quality attributes and the questions leading to the quality measurement of two architectures, was reviewed and adjusted by both IBM employees and supervisors from the University of Lund, the model is regarded to maintain a high level of inner validity. After all, the model measures what it is intended to, i.e. the level of quality in both SOAs and non-SOAs. *(Patel & Davidson, 1994; Miles & Huberman, 1997)*

To what extent the answers from the respondents can be regarded as truthfully is hard to determine, but since UBS is interested in finding out to what level of quality the new maintains, it can be assumed that the answers have been given accordingly. This can be supported by he results being received, i.e. as the answers are plausible it can be assumed that also the external validity is good. *(Patel & Davidson, 1994; Miles & Huberman, 1997)*

What might influence the level of outer validity negatively is the fact that the five respondents gathered their answers by themselves on one questionnaire sheet, without specifying the exact approach. In other words, to what extent the gathering was fulfilled correctly, i.e. all answers being equally considered, is uncertain. Not even the fact that this gathering was completed during discussion with all concerned present makes this approach more correct. Thus, the level of reliability in the research model might in fact be low, even though the respondents were not influenced by the evaluator. *(Patel & Davidson, 1994; Miles & Huberman, 1997)*

THEORETICAL FRAMEWORK

AT FIRST, THE CONSTANTLY RETURNING TERMS ARCHITECTURE, SOA, QUALITY, AS WELL AS CLOSELY RELATED TERMS, WILL BE HIGHLIGHTED. HAVING STATED THAT, THE CHAPTER CLOSE-UP WILL EVOLVE AROUND THE FACTORS MAKING UP THE RESAERCH MODEL.

One of the basic economic problems is **scarcity**, which means that the desire of getting exactly what one wants is limited by the amount of resources available. Human needs are said to be virtually unlimited, which means that their wants and needs to consume will never vanish, irrespective of the existing limiting factors and resources. (*Gillspie, 1999*) This constantly evolving and never vanishing desire is, according to Young & Biz/ed (1996-2006), due to the fact that goods wear out and need to be replaced, new products become available and create interest and finally that people get tired with what they have.

Furthermore, this economical problem does not apply for individuals only, but also for larger groups such as businesses. A business will, for example, most certainly consider new or restructured system structures if their technical support breaks down, turns out to be inefficient or if the commercials for new technical advancements succeed in generating the need for the presented products.

Due to the fact that wants are unlimited and the resources of a business (or individual) are limited at some point, choices have to be made. Thus, if a business is being interested in implementing and/or rearranging a **system** – "A collection of components organized to accomplish a specific function or set of functions." *(IEEE, 1990, p. 73)* – the return on investment (ROI) and available resources, such as finance, people skills, time etc., have to be carefully considered. Moreover, this considering and planning has to be aligned with the overall business strategy or more precisely the factors that are vital for the successfulness of the strategy. Rockart (1979) describes these factors as being critical for the success of the company, i.e. if these so-called critical success factors (**CSFs**) are not managed properly the business is likely to experience fatal consequences. *(Huotari & Wilson, 2001)* In other words, a business investing in a new system has to go through many different aspects that all influence the outcome of the choice being made.

ARCHITECTURE

Besides having knowledge about what should be invested to improve the technical infrastructure within the business, customers usually also have a good idea about what features the future system is to contain. These features are part of what software engineers call **requirements specifications:**

> "A document that specifies the requirements for a system or component."
> *(IEEE, 1990, p. 63)*

The **requirements** are usually specified by the business itself or with the assistance of consultants, represent the customer's demands and provide the foundation of the system being implemented:

> "A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. […]" *(IEEE, 1990, p. 62)*

> "A requirement describes a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document." *(Kozaczynski, 2002, p. 1)*

On behalf of the requirements, engineers use a systematic approach to structure possible relations and information into the so called **architecture**:

> "The organizational structure of a system or component." *(IEEE, 1990, p. 10).*

For smaller systems the structuring of an architectural design might not be necessary, but for larger ones it is essential to, amongst others, understand the range and complexity of the system structure, to embed IT into already existing systems, to locate geographically distributed hardware components and to estimate the effect of modifications. *(Glinz, 2001, 2003)*

SERVICE-ORIENTED ARCHITECTURE (SOA)

Bieberstein et al. (2006) address the fact that the companies of today "[…] no longer require a high degree of optimal performance for repetitive processes."*(Bieberstein et.al., 2006, p. 16).* The focus of today instead lies on their ability to reduce the time to market, as well as supporting their customers "[…] with flexible, well-suited solutions appropriate to their need." (*Bieberstein et.al., 2006, p. 16*) This demand of better integrated solutions, together with increased services shows the evolution from product-orientation to service-orientation. SOA, service-oriented architecture, is an architecture taking this evolution into consideration by having both a technical and a business oriented perspective, as well as basing the complete fundament on services.

Out of a **business perspective**, SOA is said to improve business agility and to maintain services being directly applicable to the existing business logic of the business:

> "A service-oriented architecture provides the flexibility to treat elements of business processes and the underlying IT infrastructure as secure, standardized components (services) that can be reused and combined to address changing business priorities." *(Bieberstein et. al., 2006, p. 4)*

The **technical perspective,** on the other hand**,** emphasizes the importance of the actual structure of the architecture, i.e. of what SOA is made of and how it works:

> "An enterprise-wide IT architecture that promotes loose coupling, reuse, and interoperability between systems." *(Bieberstein et. al., 2006, p. 4)*

> "An application architecture in which all functions or services are defined using a description language and have callable interfaces that are called to perform business processes. Each interaction is independent of each and every other interaction and the interconnect protocols of the communicating devices. Because interfaces are platform independent, a client can use the service from any device using any operating system in any language." *(Bieberstein et. al., 2006, p. 4-5)*

Summing up this in a general overview, being useful through the following sections of SOA, factors such as front-end, Enterprise Service Bus (ESB), service, service repository, interface, business logic and data have to be regarded. All these can be presented in the following

hierarchal figure, which due to its simple structure is not regarding that services can be both requestors and providers (see section *Service*):

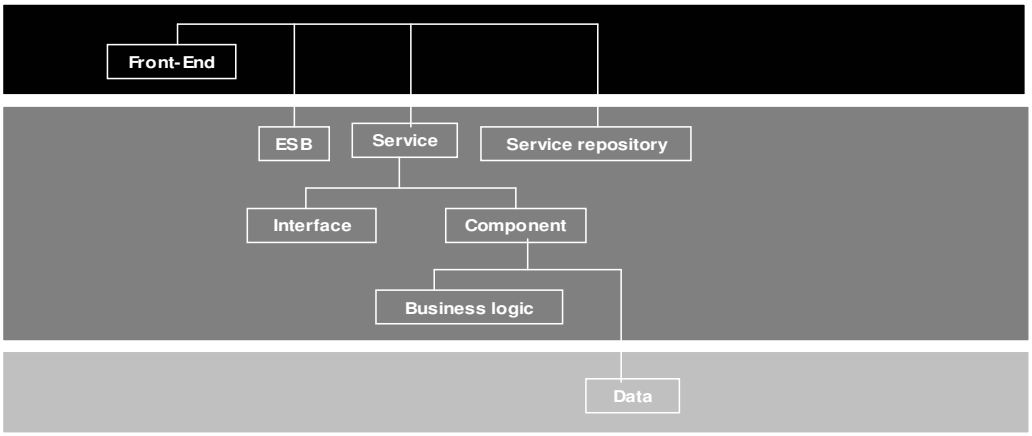


Figure 20: The components of SOA, only considering service providing aspect.
(Inspired by: Wikipedia, 2006, *http://en.wikipedia.org/wiki/Image:SOA_Elements.png*)

CLIENT/SERVER ARCHITECTURE, FRONT-END

On a general basis, SOA is best descried as a client/server architecture. In fact, Schulte (2002) affirms this statement by claiming that SOAs are usually built upon two or more tiers. The tier maintaining the graphical user interface (GUI) and representing the so called Front-End is called Presentation-tier, while the one presenting the components with the business logic is called Business logic-tier. The latter one provides services to the clients in the Presentation tier or Business logic-tier, all depending on where the requestor is situated.

Finally, in a three-tier architecture there is also the Data-tier or Back-End, where data is stored in and retrieved from, for example, databases. (*Hammerschall, 2005*) In other words, SOA is the fundament for systems with "[…] software services and software consumers (also known as clients or services requestors)."(*Natis & Schulte, 2003, p.* 1)

Figure 21: A three-tier client/server or service requestor/service provider architecture, disregarding that requestors might also occur on the Business logic-tier. (Inspired by: *Hammerschall, 2005*)

However, one might also challenge the statement of SOA being client/server architecture. After all, not only one server acts as provider, but normally several, where each and every one of them might be a service that is available to all possible clients or requestors. *(Dahan, 2004)* To go deeper into that statement, one has to take a closer look at the Business logic-tier. In this tier, SOA has a special architectural structure originating from the available services and the ESB:

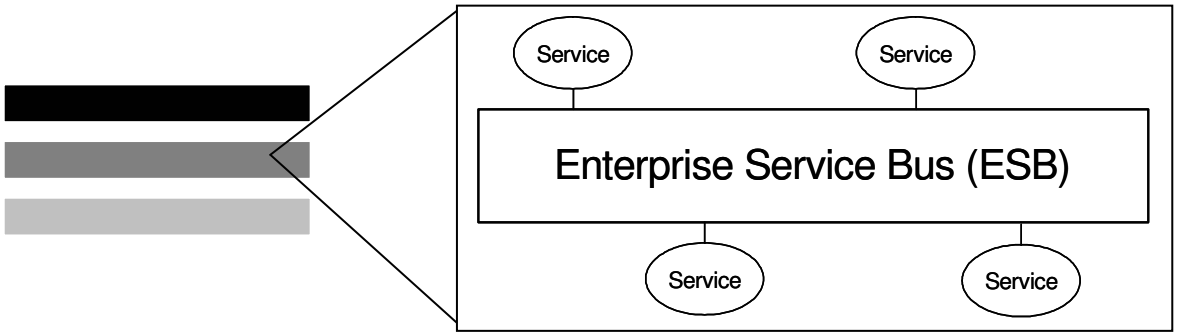


Figure 22: ESB with its services as being part of the Business logic-tier.

Up until recently integration services were implemented with point-to-point messaging systems, such as IBM's WebSphere Message Queuing (MQ) (formerly MQ Series) and Microsoft Message Queuing (MSMQ). **ESB**, as a middleware, has extended this integrability by combining the messaging systems and thus creating an exchange between a service provider and service consumer in terms of classical messaging, as in EAIs, or through objects, as in, for example, in Common Object Requester Broker Architectures (CORBAs) or Web Services. (*Wong-Bushby et.al., 2006*)

As Chappell (2004) states, the ESB allows data from one application to be sent to any other application without any advance knowledge. More precisely, spread services of SOA can be accessed by other services by simply calling the available service-interface (*Hasselbring, 2006)*, which in turn provides transparency and thus eases the interaction.

As ESB is not regarded as a major contributor to the overall aim of this thesis, other ESB features ensuring that disparate technologies of all kinds to work well together, as e.g. infrastructure services such as transport, quality-of-service-based routing and gateway services (*Bieberstein et.al., 2006*), are not mentioned any further.

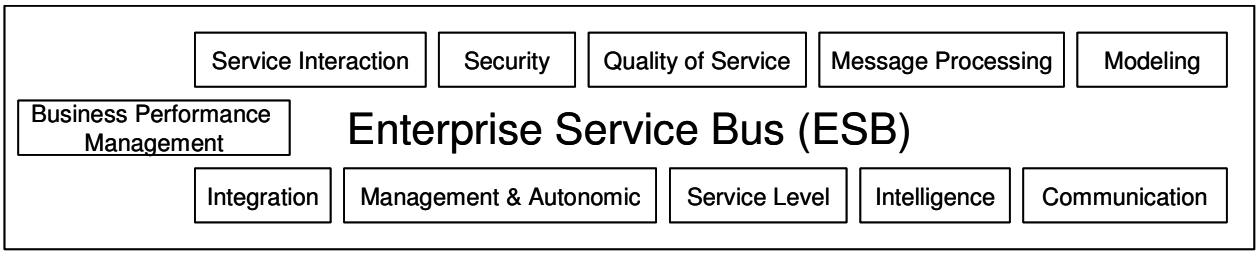


Figure 23: All parts of an ESB (*Keen et.al., 2004, p. 45*)

Assuming that the ESB manages its interaction with Web Services, the industry standards being used are:

- eXtensible Markup Language (**XML**); is a descriptive language that is mainly used for the description and exchange between complex data structures. XML is administrated by the W3C. (*Hammerschall, 2005*)

  > "A general-purpose markup language developed by the W3C for the definition, transmission, validation, and interpretation of data/information between applications and between organisations. The extensibility allows the creation of specialized markup languages and domain definitions with their own customized tags by using a formal grammar and vocabulary (called an XSD)." *(Bieberstein et.al., 2006, p. 207)*

- **SOAP**; is a middleware protocol that is based on a transport protocol, such as for example Hyper Text Transfer Protocol (HTTP) or Simple Mail Transport Protocol (SMTP), and uses this structure to transmit packages in XML format. SOAP was initially the acronym for Simple Object Access Protocol, but since the protocol turned out to be neither simple nor object oriented, the protocol is today simply called SOAP and is administrated by the World Wide Web Consortium (W3C). (*Hammerschall, 2005*)

  > "A XML-based messaging protocol maintained by W3C that is used to encode the information in Web Service request and response messages before sending them over a network. SOAP messages are independent of any operating system or protocol and can be transported using a variety of protocols, using HTTP and Java Message Service (JMS)." *(Bieberstein et. al., 2006, p. 215)*

- Web Service Description Language (**WSDL**); Besides a protocol like SOAP, a middleware also needs an interface language. For Web Services, WSDL is this interface language that defines the interface using XML. WSDL is also administrated by the W3C. (*Hammerschall, 2005*)

  > "A standard language for defining a Web Service description. It uses XML and XSD to describe the port type and its operations, the message formats, ad the protocol binding." *(Bieberstein et.al., 2006, p. 217*)

- Universal Description, Discovery and Integration (**UDDI**): A Web Service does not only need access protocols and interface descriptions, but also a registry for publications. The UDDI describes the registry interface for Web Services. By doing so the information, such as for example the name, of all distributed services is registered at one place.

  > "An Organization for the Advancement of Structured Information Standards (OASIS) standard for a platform-independent, XML-based registry to publish and discover network-based software components and services." *(Bieberstein et.al., 2006, p. 216)*

SERVICE REPOSITORY

Together with the middleware action, such as service interaction, the Business logic-tier also maintains **service repositories** or directories (see UDDI in section *Enterprise Service Bus (ESB)*) that store meta data, i.e. the meta data of published services.

> "[…] repository is similar to a place where construction goods are stored and can be instantly retrieved for use when needed." *(Heineman & Councill, 2001, p.25)*

In other words, instead of service requestors going directly to potential services, the service repository is contacted and retrieved. To manage all the requests arriving at the repository, SOAs use service brokers, which put all requests into queues. (*Hammerschall, 2005*):

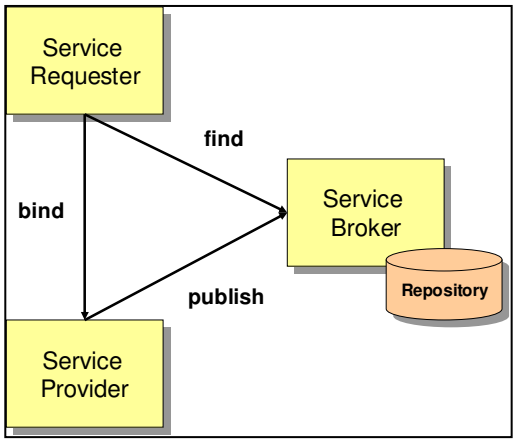
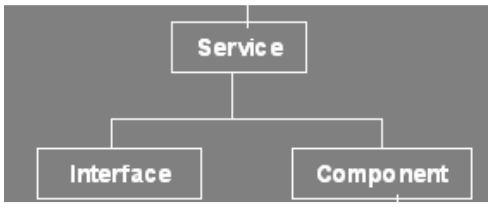


Figure 24: The process between service requestors and providers through services brokers to a repository. (*Ebner, 2003, p. 41*)

SERVICE



When discussing what a service actually is about, generally the economic definition is mentioned at first. In other words, a service is said to be equivalent to a non-material good, as for instance the guarantee of being able to call a service centre 24h a day when purchasing a specific product. Services, however, might also be used in a technical sense. It has, for example, already been mentioned that services are exchanged between requestors and providers over the ESB through interfaces. Furthermore, it as been realized that services have to be the most essential units of the complete service-oriented architecture, due the name of the architecture. Having stated this and considering the definitions of Bieberstein et.al. (2006; 2005), Natis & Schulte (2003), Lager (2006) and Keen et. al. (2004), IT related services are shortly and on a general basis to be defined in terms of:

- Encapsulated business **components** that are exchanged between service requestors (usually on the Presentation-tier) and providers (Presentation-tier or Business-logic tier) through defined implementation independent **interfaces**:

> "An application component deployed on network-accessible platforms hosted by the service provider. Its interface is described by a service description to be invoked by or to interact with a requester." *(Bieberstein et. al., 2006, p. 214)*

> "At design time, a service is an encapsulated business software component that is rendered as a pair of separately defined elements — service interface and service implementation." *(Natis & Schulte, 2003,p. 1)*

> "Miko Matsumura, vice president of technology standards at Infravio, offers this definition: A service is a network-accessible function, abstracted behind an interface." *(Lager, 2006, p. 21)*

- A **collection of end points**, i.e. being composition of several components (see *Figure 25*, section *Components*):

> "A service is a collection of related endpoints." *(Bieberstein et. al., 2005, p. 150)*

- **Reusability**, i.e. dynamic services are at any time intended to be reused, or at least accessed:

> "[…] Services encapsulate reusable business function." *(Keen et. al., 2004, p. 37)*

- The lacking dependencies between consumers and providers (**loosely coupling**). More precisely the consumers should not be dependant on any information about where the service provider is located, on what platform the services is implemented and with what language the service is programmed:

> "[…] Services are loosely bound and invoked through communication protocols that stress location transparency and interoperability. […] ." *(Keen et. al., 2004, p. 37)*

- Communication feasibility to other services through the implemented **granularity**, i.e. the level at which the service is created. Some claim that services should be coarse grained while others respond by emphasizing the essence of fine grained services. According to Bieberstein et al. (2006), choice depends on the analysis and design for SOA solutions. Coarse grained solutions might for example be better than fine grained solutions in SOA network capacity planning where the services have to present less detailed parts of the business.

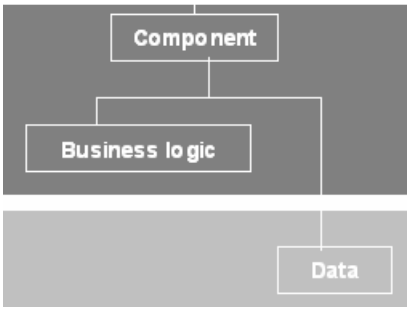
A final discussion that simply has to be included, even though it is only limited, when mentioning services is the one about Web Services. More precisely, it is about to what extent services really are to be used synonymously with today's so frequently used concept **Web Services**:

> "A family of technologies that consist of specifications, protocols, and industry-based standards that are used by heterogeneous applications to communicate, collaborate, and exchange information among themselves in a secure, reliable, and interoperable manner. *(Bieberstein et.al., 2006, p. 217)*

Considering that Web Services are services that are exchanged over the Internet with specific protocols (see section *Enterprise Service Bus (ESB)),* in contrast to services, it quickly

becomes clear that it is in fact even incorrect to use these concepts synonymously. However, since Web Services are a good way of showing the uniqueness of SOA, this thesis will use the concepts of services and Web Services equivalently.

COMPONENTS



Components, might just like services occur both in the Presentation-tier and in the Business logic-tier. Where the components are situated depends on if they are requestors or providers. Providing components, for example, will most likely be situated at the Business logic-tier. Requestors, on the other hand, can be both on the Presentation-tier and the Business logic-tier. Since components not only provide encapsulated business logic, but also, for example, access to old applications, also the Data-tier has to be mentioned in combination with components.

In summary, components encapsulate business logic, provide data access and expose one or more services with interfaces to service requestors over the ESB:

> " - […] is a unit of independent deployment;
> - is a unit of third-party composition;
> - has no (externally) observable state." (*Szyperski, 1999, 2002, p.36*)

> "A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard." (*Heineman & Councill, 2001, p. 7*)

The following figure pictures three possible relationships between components and services, i.e. a component with on service only, a component that exposes three services and a service being composed of two components:

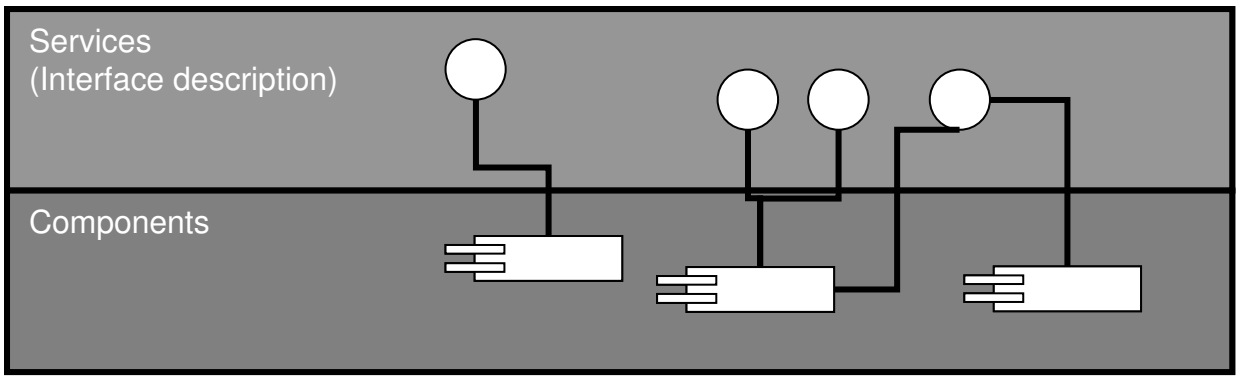


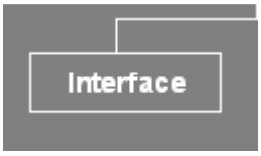
Figure 25: Possible service and component relationships.

Szyperski (1999, 2000) mentions the interchangeable use between components and object, due to similar features such as making services available through interfaces and creating interaction to other components/objects through patterns and frameworks. However, taking a closer look at the real definition of **objects** the difference becomes clear:

" - […] is a unit of instantiation, it has a unique identity;
- may have state and this can be extremely observable;
- encapsulates its state and behavior." (*Szyperski, 1999, 2000, p. 37*)

Thus, components, in contrast to objects, have no actual need of only containing classes or even at all. Instead components might contain "[…] traditional procedures and even have global (static) variables, or it may be realized in its entirety using a functional programming approach, or using assembly language, or any other approach."(*Szyperski, 1999, 2000, p. 38*)

INTERFACE



The **interfaces** are the "contracts" creating transparency, maintaining all the information needed (information hiding) to symbolize one specific service, as well as gathering all component end-points for system independency:

> "(1) A shared boundary across which information is passed.
> (2) A hardware or software component that connects two or more other components for the purpose of passing information from one to the other.
> (3) To connect two or more components for the purpose of passing information from one to the other." (*IEEE, 1990, p. 41)*

> "Interfaces are provided to wrap service endpoints to provide a system-independent architecture to promote cross-industry communication." *(Keen et.al., 2004, p. 25)*

In other words, services are invoked by service requesters that are unaware of the details about the service implementation. This lack of dependencies between consumer and provider is called **loose coupling** and involves hiding of details within the areas of **location, platform**, and **language** and in some cases even information about the service provider. In other words, services being distributed on different platforms (for instance .NET or Java 2 Platform Enterprise Edition (J2EE)) with different languages (for example Java, Cobol or C++) communicate with each other through the earlier mentioned ESB with the help of different standards. (*Bieberstein et.al, 2006*)

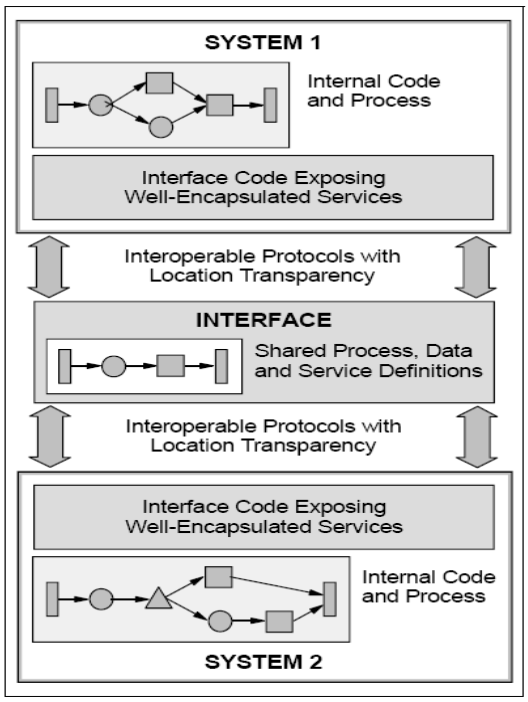


Figure 26: Interfaces creating interaction. (*Keen et.al., 2004, p.56*)

Decoupling, or more precisely the ability of offering service consumers distributed services, creates greater security challenges than architectures focusing on centralized and non-Web based data sources. For SOA this means an additional **security** – "Zustand des Nichtvorhanden- oder Geschuetztseins vor Bedrohung und Risiken" (*Opplinger, 2005*) - aspect, which is managed with Web Service security (WS-Security). WS-Security is made up of several units, handling different aspects in accordance to demand, and enables the user to apply "[…] XML security techniques to authenticate and secure message exchanges between a Web Service requestor and a Web Service provider. It uses signatures and encryption placed on a message and security tokens bound to the messages." *(Bieberstein et.al., 2006, p. 148)*

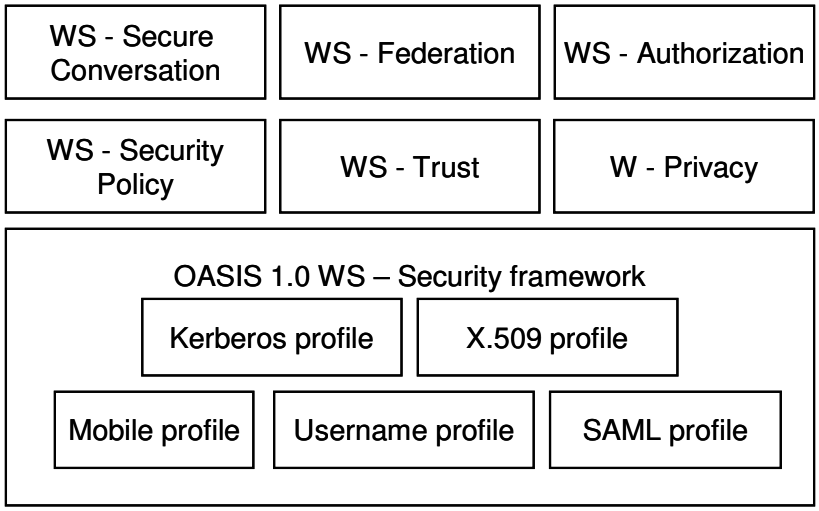


Figure 27: WS-Security with its units, covering different security aspects.
(IBM Software University, 2006, p. 14)

SOA RESUME

After having presented some introducing general definitions of SOA, as well as some detailed information about different attributes of the architecture, it might seem difficult to put forward a general definition. In attempt of avoiding a too general phrasing, one would lean on the definitions of Keen et. al. (2004) and Bieberstein et. al. (2006):

> "SOAs consist of services that are defined by explicit, implementation independent interfaces. They are loosely bound and invoked through communication protocols that stress location transparency and interoperability. Services encapsulate reusable business function." *(Keen et. al., 2004, p. 103)*

> "A service-oriented architecture is a framework for integrating business processes and supporting IT infrastructure as secure, standardized components – services – that can be reused and combined to address changing business priorities."
> *(Bieberstein et. al., 2006, p.5)*

Putting these definitions into an overall picture, as well as gathering the attributes being discussed in earlier chapters about SOA, this figure covers them all:

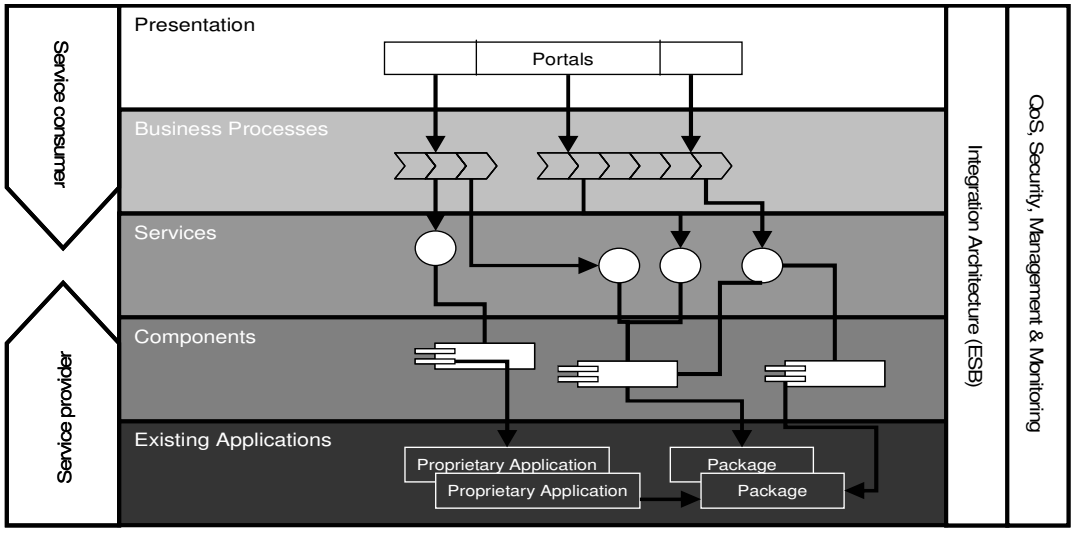


Figure 28: Overview of service consumer and service provider interaction. *(Peisl, p. 9)*

QUALITY

As mentioned in the introducing words of the *Theoretical framework*, an inappropriate managing of success factors, such as product development, distribution, advertisement within respective industry and customer satisfaction *(Huotari & Wilson, 2001)*, is likely to hinder a business from being or becoming successful. In attempt of avoiding this potential risk of failure, companies seek to identify and eliminate possible defects or mistakes within the business. One approach being frequently used in the industry today is the industry standard Six Sigma, being introduced by Motorola Inc in the late 1980s. *(Snee, 2004; Kan, 2003)*

This methodology of managing defect business processes is defined as a "[…] stringent level of quality." (Kan, 2004, p. 66). Having this in mind, it becomes more and more clear, what companies actually strive for. After all, a higher level of internal quality provides the opportunity of also achieving a higher level of external, i.e. towards customers, members, stakeholders etc., quality. This in turn, generates a competitive advantage over competitors

and can be regarded as a great business asset. Ortega et. al. (2003) extend this quality discussion by adding the perspective of customer needs:

> "Quality is currently considered one of the main assets with which a firm can enhance its competitive global position. This is one reason why quality has become essential for ensuring that a company's products and processes meet customers' needs." (*Ortega et. al., 2003, p. 219*)

Even though UBS per se provides services towards customers, UBS also can be seen as a service consumer and is therefore in need of getting the so-called customer needs satisfied. The chapter *Research area – UBS* addresses this fact by giving a summary of the architectural change that has been taking place, i.e. the implementation of the service-oriented architecture. To sum up this reasoning, with a new IT architecture UBS is expecting a certain outcome, due to the specified requirements and formulated needs. To what extent this outcome symbolizes the earlier discussion of higher level of quality assumes the usage metrics.

To specify the needed quality metrics in combination with SOAs it first has to be absolutely clear what in fact **quality** is, in combination with SOAs. Even though the concept is said to be difficult to define, describe and understand *(Bratthall & Wohlin, 2000; Kan, 2003)*, the definitions of ISO (1986), IEEE (1990), Glinz (2001, 2003) and Kan (2003) are regarded to cover the aspects being considered in this thesis. By using these definitions, it is not neglected that all three definitions are to describe software quality, but as SOA can be seen as the foundation for software, see *Figure 28* in section *SOA Resume*, this is not regarded as a limiting factor.

Kan (2003), ISO (1986), Glinz (2001, 2003) and IEEE (1990) address the aspects of to what extent specified requirements, as well as customer needs or expectations, are met and how this is measured:

> "(1) The degree to which a system, component, or process meets specified requirements. (2) The degree to which a system, component, or process meets customer or user needs or expectations." *(IEEE, 1990, p. 60*)

> "[…] quality can, and should, be operationally defined, measured, monitored, managed, and improved." *(Kan, 2003, p.2)*

NON-FUNCTIONAL AND FUNCTIONAL REQUIREMENTS

As quality has a strong correlation to requirements it is important to address the requirements being frequently discussed in the context of IT. According to Glinz (2001, 2003) specified requirements for a product are normally divided into functional and non-functional requirements. Even though these two kinds of requirements are difficult to separate, the **functional requirements**, focusing on to what extent the product actually does what it is expected to do, are the requirements that usually receive the greatest attention and are thus normally also fulfilled. After all, if the system does not provide the system user with the requested functionality the IT consumer very quickly address this issue.

> "A requirement that specifies a function that a system or system component must be able to perform." *(IEEE, 1990, p. 35)*

Non-functional requirements, on the other hand**,** are said to be the constraints of the system's functions or tasks and are thus less obvious and harder to identify by the IT consumer.

> "Nicht-funktionale Anforderungen – Anforderungen an die Umstände, unter denen die geforderte Funktionalität zu erbringen ist." (*Glinz, 2001, 2003, p. 12-1*)

Hence, these non-functional requirements or so called "-ilities" receive less attention and thus become more critical. Despite this, or more likely, because of this, several of the non-functional attributes describing the technical aspects of a system have been defined by multiple different organisations and companies, such as The International Standards Organization (ISO) and International Electrotechnical Commission (IEC) (1994) with the report **ISO/IEC 9126** (1994)**,** IBM with **CUPRIMDSO** (*Bencher, 1994*) and Hewlett-Packard with **FURPS** (*Grady & Caswell, 1987*).

The business perspective in the non-functional requirements, however, has up until now been both neglected and omitted when mentioning technical quality attributes. None of the just mentioned standards and models alludes, as Bass et al. (1998) for example calls it, "Not observable via execution" (*Bass et. al., 1998, p. 76*), where one instead shows interest for the integration of the system, the cost of development and time to market.

QUALITY ATTRIBUTES OF THE SOA QUALITY EVALUATION MODEL

As stated by Ortega et.al. (2003), not only defining attributes (see previous chapter) have gained in interest, but also creating models to alleviate an overview and correlations of quality. The models of Boehm (1978) and McGall (1997) are two of the most frequently mentioned. The available models, however, do not consider SOA quality attributes, but rather software quality or other SOA aspects (see section *Model structure Of the SOA quality evaluation model*).

In correlation with the chapter *Methodological approach*, showing the approach for the attribute selection, this chapter will present the theoretical background of attributes leading to the determination of a SOA quality level. The system standards discussed in the previous section of *Non-functional and functional requirements*, are used as a foundation for the attributes in this so-called SOA Quality Evaluation Model, but to gain the most applicable attributes, describing SOA features, these standards are both refined and replaced by other attributes.

The overall structure of the SOA Quality Evaluation Model is based on both business and technical oriented aspects, as this is considered to be one of the main strengths of SOA, i.e. to combine IT and business. Hence, in accordance with the discussions held in the sections *Service-oriented architecture (SOA)* and *The approach*
Having defined the strategy to be followed and the method to be applied, the overall preparations for the research, to be conducted, had to be made. In other words, attributes had to be selected, model structures had to be evaluated and a questionnaire had to be created.

Gathering SOA Quality attributes, the quality attributes have been subordinated to either the section *Technical Perspective* or *Business perspective*.

TECHNICAL PERSPECTIVE

QUALITY ATTRIBUTE (1): MODIFIABILITY

Very often, the concepts of modifiability and **maintainability** are used synonymously. In contrast to this, Bass et. al. (1998) states that some authors insist on keeping these two concepts apart and using them differently along with the type of change that is being made. In other words, modifiability is to be used when the change involves a modification of attributes within an architecture:

> "Eine Menge von Merkmalen, die sich beziehen auf den Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist." *(ISO/IES, 1994, p. 4)*

Maintainability is more applicable in the context of simply maintaining these attributes. Due to the aim of using a concept that points out changeability, modifiability is defined to be the first technical quality attribute in the SOA Quality Evaluation Model.

According to Bass et. al. (1998) modifiability can be regarded as the attribute with the closest connection to architecture. This, mainly because the attribute focuses on to what extent certain attributes within the architecture can be modified. In other words, modifiability is not about the change of the overall architecture, but rather the change of processes, products, technologies, behavior (rules) etc.:

- *Extending or changing capabilities,* i.e. new features are added and/or old ones are being repaired or simply enhanced.
- *Deleting unwanted capabilities* involves reducing the range of the system by deleting functions that are not needed.
- *Adapting to new operating environments* mostly concerns the introduction of new hardware, but also different business conditions.
- *Restructuring* concerns, for example, how to change the architecture from object-oriented (OO) to component-oriented.
  (Bass et. al., 1998)

In accordance with section *Interface*, SOA achieves this through the modularization, encapsulation, loose coupling of components and configurable applications. Hence, the code of the system should not have to be changed, but only a reconfiguration of architectural objects should be necessary. In other words, by configuring or managing through for example an IBM Tivoli Monitoring Tool, it should be possible to simply choose the affected components and services and rearrange, add or delete these without actually affecting the overall architectural structure. (*Bieberstein et.al., 2006)*

QUALITY ATTRIBUTE (2): PORTABILITY

The attribute modifiability evaluates to what extent it is possible to modify attributes of the architecture without affecting the overall architectural structure. Portability on the other hand, evaluates if the overall architecture can be moved to another environment, if it is adaptable and replaceable:

> "[…] the ability of the system to run under different computing environments. These environments can be hardware, software, or a combination of the two."
> *(Bass et.al., 1998, p. 83)*

The environment mentioned could for instance either concern the ability of changing platforms or the ability to move the system to completely new areas, for example other countries and cultures. A world wide company in Switzerland, for example, has recently implemented a new architecture and now wants to apply this architecture with the same conditions the U.S., where a separate division of the company is situated. To succeed, the architecture has to be portable. *(Krüsemann, 2006)*

Another aspect of portability is cultural internationalization. Using the above example again, this would mean that the architecture has to consider the cultural differences between U.S. and Switzerland. North-Americans for example, have other rules, regulations and legislatives that have to be taken into consideration and/or specific preferences for system usage and of the products being involved. More precisely this would mean that North-Americans, for example, still prefer the traditional pay slips, while the Europeans frequently use E-banking and/or that stores are preferred, instead of for example the internet. Moreover, of course, languages are another aspect. A system that is only available in German is of no use for Americans. In other words, if all these cultural aspects are not considered users will be reluctant to use the system, no matter how good the system might be in other countries. *(Krüsemann, 2006)*

As stated in the section *Interface*, one of the strengths of SOA evolves around platform independency. In other words, in the case of SOA, platform specific information is being encapsulated and hidden behind an abstract interface, offering portability in terms of transparency to the system being based upon SOA:

> "The encapsulation of platform-specific considerations in an architecture typically takes the form of *portability layer*, a set of software services that gives application software an abstract interface to its environment." *(Bass et. al., 1998, p. 83)*

Thus the system will be adaptable to different kinds of environments without influencing the core of the already existing system:

> "[…] the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered." *(Centre of Software Engineering (Essi-Scope), 2003, http://www.cse.dcu.ie/essiscope/sm2/9126ref.html)*

The explanation of portability in terms of replaceability would simply mean that existing parts of the system, as e.g. the hardware, might be replaced without affecting the system on the whole:

> "[..] bear on opportunity and effort using it in the place of specified other software in the environment of that software." *(Centre of Software Engineering (Essi-Scope), 2003, http://www.cse.dcu.ie/essiscope/sm2/9126ref.html)*

QUALITY ATTRIBUTE (3): REUSABILITY

According to Bass et.al. (1998) reusability is an attribute which questions to what extent different system components can be reused, either within the same or in another system. Reused in the sense that the components do not have to go through any changes, but can simply be used the way they are and have been defined:

> "*Reusability* is usually taken to mean designing a system so that the system's structure or some of its components can be reused again in future applications." *(Bass et. al., 1998, p. 84)*

Having the definition from Bass et.al. (1998) in mind, it becomes quite obvious that there is an indirect relationship between modifiability and reusability. More precisely, modifiability benefits from reusability. Changes or modifications, for example, might turn out to be either efficiently or simply inefficiently conducted, all depending on whether or not the components are loosely or strongly coupled.

Heineman & Council (2001) state that developers during several years have been discussing reusability, but without really having been able to achieve a significant degree. With the introduction of reusable component and service technologies discussions instead moved towards topics such as custom-made vs. standardized solutions, where both parties have benefits and limitations:

Custom-made solutions:	**Standardized solutions:**
- are built from scratch - are expensive and difficult to build - are not available for the customer until it is built - are expensive to maintain and develop - the return on the investment arrives relatively late	- are badly aligned to the business - do not give competitive advantages - are difficult to maintain and develop, if the cost advantages are to be kept

Figure 29: Drawbacks with custom-made and standardizes systems. (*Szyperski, 1999, 2002*)

Within this discussion Szyperski's (1999, 2002) presents a theory, where the concept of components is to combine the architectural and financial drawbacks of both standardized and custom-made solutions, as components per se are standardized products with the opportunity of customization. More precisely, Szyperski (1999, 2002) states that components are architectural parts that have the ability of inheriting advantages such as the lower development and maintenance costs, stability (e.g. reduced error rate) and quality from the standardized solutions and the custom made shape, efficiency, and adjustability of the custom-made solutions. In *Figure 30* this theory is presented visually, by showing where components would be situated (the dotted square) in proportion to a custom-made (to the left) and a standardized (to the right) solution:

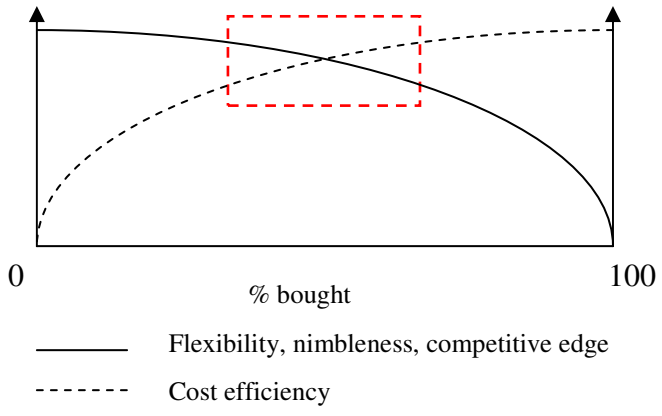


Figure 30: The difference between make-all and buy-all. (*Szyperski, 1999, 2002, p.6*)

Applying Szyperskis's (1999, 2002) theory on SOAs assumes the close interaction between components and services described in the section *Components*. Moreover, the fact that services per se are said to be reusable (see section *Service*), supports the reason for regarding the theory as being applicable to SOAs even more. Hence, both services and components can be allocated in the centre of the graph in Figure 30.

Finally, one could challenge the statement of SOA components and services being less expensive to develop. After all, at this point in time still a lot of services have to be created and defined before they eventually can be reused. Hence, developers currently have to invest more effort, money and time into the development of components than what can be expected once the overall register with components is created:

> "[…] to standardize the *connections* among all those components so that they work the same way everywhere without requiring additional, customized programming, which is costly and can prevent reuse." *(Bieberstein et. al., 2006, p. 19)*

QUALITY ATTRIBUTE (4): INTEGRABILITY

Integrability is the quality attribute covering everything concerning integration between two or more components and services of a system:

> "Integrability is the ability to make the separately developed components of the system work correctly together." (*Bass et.al., 1998, p. 84*)

Together with this concept usually also **interoperability** is mentioned, to highlight the possibility that not only separate components need to be integrated but also, for example, groups of parts with other old or new systems:

> "Interoperability measures the ability of a group of parts (constituting a system) to work with another system." (*Bass et.al., 1998, p. 85*)

To be able to cover the essence of both single and grouped components in an appropriate way derived from the service and component description in section *Components*, the concepts integrability and interoperability are both gathered in this quality attribute, called integrability.

According to Bass et.al. (1998) the ability of integrating loosely-coupled components or services depends on the external complexity of the components/services, the interaction mechanisms, the protocols used, as well as all other issues being typical for each architectural level as in for instance the Open Systems Interconnection (OSI) Reference Model.

Furthermore, integrability considers the interface of the component/service, i.e. how well and completely defined the interfaces of the belonging components/services are. In other words, this attribute includes everything that was described as being a part of the Business logic-tier (see section *Client/Server Architecture, Front-End*), including the interaction mechanism ESB, the protocol SOAP, the interface language WSDL, the directory UDDI, interfaces and components/services.

QUALITY ATTRIBUTE (5): SECURITY

As mentioned in section *Interface*, the security aspect in combination with SOAs become especially important due to the introduction of Web Services. Surely, a SOA does not have to

use Web Services, but still this is a very SOA specific feature (see discussion in section *Service*) and is therefore regarded as an essential part of a SOA Quality Evaluation Model.

Disregarding some of the security specific term, this quality attribute is mainly about providing architectures with prevention of unauthorized access, both accidental and deliberate:

> "Security is a measure of the system's ability to resist unauthorized attempts at usage and denial of service while still providing its services to legitimate users."
> *(Bass et.al., 1998, p. 80)*

Changing the perspective and going into somewhat more specific security concepts, the aspects being pointed out in the section *Interface*s indicate what WS-Security can do to support Web Services:

- **Authentication**, i.e. the verification of a given identity by:
  - "Having something" like a key, ticket, membership card etc.
  - "Knowing something" like a personal PIN, password etc.
  - "Being someone" like facial features, DNA-tests, finger prints etc.
  - "Being somewhere" like telephone number recognizer, verification systems for Internet Protocol (IP)-addresses

  (Opplinger, 2005)

- **Authorization** and **Access control**, i.e. the process of deciding what actions the concerned entity will be allowed to perform. (*Bieberstein et.al., 2006*)

- **Firewalls**, which are walls between at least two networks, allow or deny access depending on given or denied authorization. (*Opplinger, 2005*)

- **Encryption** placed on a message, is "[…] the process of converting information from one format to another using a mathematic transformation […]" (*Bieberstein et.al., 2006, p. 142*) and thus making it unreadable without special knowledge. More precisely encryption involves the message and a key and the output generated with these inputs. Before the receiver can read the secured message he/she has to use another key and decrypt the encrypted message. Another approach involving encryption is the so called Public Key Infrastructure (PKI), where both the provider and requestor have a key of their own and "[…] exchange the corresponding public key certificates with those partners with whom they wish to establish trust." (*Bieberstein et.al., 2006, p. 142*). On example of PKIs is the digital signing of messages.

Finally, one should not forget that all these security aspects are just as important without Web Services. A simple example would be the accessing of a database. Since most likely not everyone, not even within the same company, should have access to a specific database, restrictions in terms of authentication, access control and authorization have to be set up.

QUALITY ATTRIBUTE (6): EFFICIENCY

The quality attribute efficiency refers to the standard definition of ISO/IEC (1994), where the level of **performance** of the system, being based on a specific architecture, and the amount of resources being used under stated conditions:

> "Eine Menge von Merkmalen, die sich beziehen auf das Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen." *(ISO/IEC 9126, 1994, p. 4)*

A somewhat definition is derived from Bass et.al. (1998), who explain performance in terms of time behavior. By doing so, it gets obvious that performance involves metrics providing results about how responsive the architecture is built, as for instance through transactions per unit time, transaction throughput, recovery time, start-up time, shut-down time and/or response time.

> "The responsiveness of the system - the time required for respond to stimuli (events) or the number of events processed in some interval of time." *(Bass et.al, 1998, p. 78)*

To go back to the first definition, it might be unclear what is actually meant with "a specific architecture". To look at this out of a SOA perspective, both a technical and a business oriented aspect are involved. The technical perspective simply involves the kinds of IT resources that are being used, i.e. only with a powerful hardware a reasonable time for efficiency can be achieved. Discussing this statement in terms of the amount of resources being used, it would for example mean deciding whether 15 or 30 servers are needed. It might seem more self-evident to achieve a higher efficiency through 30 servers, due to parallel processing, but on the other hand, 30 distributed servers will demand a higher level of management and thus not necessarily increase efficiency. Hence, the architectural structure plays a great role in terms of kind of IT present, the interaction between different architectural parts, process synchronization, queue size of requests and latency. *(Bass et.al., 1998)*

The other architectural aspect, affecting the efficiency, is the way the business is structured, i.e. how efficiently the business processes are. As long as these processes are not optimized, the system will be seen as inefficient. More about business processes is discussed in section *Quality Attribute (10): Business Flexibility*.

Finally, taking a silo based and service-oriented architecture as an example, evidence of inefficiency and efficiency becomes more obvious, i.e. the silos and their vertical executing and subsequent functions in contrast to the service and component oriented architecture.

QUALITY ATTRIBUTE (7): SCALABILITY

Scalability, according to Harishankar (2001), is "[…] the capability of a system/component to adapt readily to a greater or lesser intensity of use, volume, or demand while still meeting business objectives […]." (Harishankar, 2001, p. 14). Bieberstein et al. (2006) and Heineman & Councill (2001) confirm this definition by stating that scalability is about considering the correlation between the total degree of throughout or performance in a system and the resources added.

As can be noticed, scalability is closely related to the quality attribute efficiency. The greatest difference between these two attributes can best be described in terms of the three aspects change of load, performance and resources. Out of this perspective, scalability involves each an every aspects individually. Efficiency on the other hand looks at all three of them at once. (Inspired by *Krüsemann*, 2006)

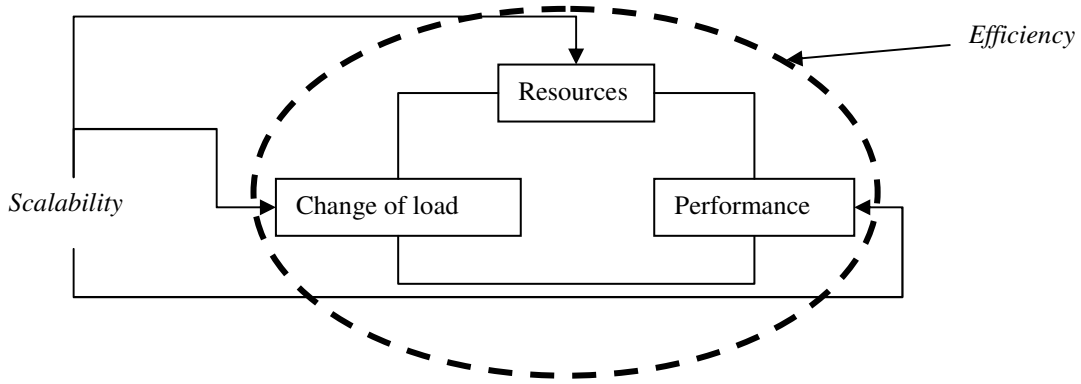


Figure 31: Comparison between scalability and efficiency.

In other words, when the system has reached the desired or at least approved efficiency, this should be possible to maintain even if the company intends to expand within different areas or within the complete company. Thus, an architecture or system that improves its performance proportionally to the adding of faster hard drives or a greater number of publicly exposed services, is said to be scalable. This is relatively difficult, since scalability not necessarily has to lead to efficiency, as already mentioned in combination with the adding of servers in section *Quality Attribute (6): Efficiency* in the.

Scalability can be divided into three different types:

- **Load scalability** – Involves the increasing or decreasing of distributed system loads, i.e. if more systems are added, the load gets heavier and the opposite if nodes are for example extracted, leading to an optimization of the architecture. (Wikipedia, 2006, *http://en.wikipedia.org/wiki/Scalability*)
- **Geographic scalability** – This kind of scalability is able to provide usefulness and usability no matter where the distributed parts of the system are situated. (Wikipedia, 2006, *http://en.wikipedia.org/wiki/Scalability*)
- **Administrative scalability** – A distributed system that is shared by several organizations and is still easy to use and manage is considered to maintain administrative scalability. (Wikipedia, 2006, *http://en.wikipedia.org/wiki/Scalability*)

According to Heineman & Councill (2001), scalability is best achieved through a middleware where "[…] caching and recycling strategies can ensure that many clients share these resources optimally between them." (*Heinemann & Councill, 2001, p. 623*) For this, SOA uses the ESB, mentioned in section *Enterprise Service Bus (ESB),* and thus should be able to provide both load, geographic and administrative scalability for service requesters and service providers.

QUALITY ATTRIBUTE (8): RELIABILITY

At most companies today, every IT system goes through a risk analysis, showing what impact a breakdown has on the overall business. In combination with estimating the maximal level of downtime indirectly also the requested level of service is stated. At UBS, for example, systems that experience breakdowns, generated through non-functional servers and/or disk crashes, have the following Standard Service Levels and maximal downtime:

Standard Service Level	Maximal Downtime per Event:
Standard: 98% availability per year	24 h
Standard +: 98.5% availability per year	12 h
Premium: 99.1% availability per year	4 h
Premium +: 99.5% availability per year	2h

Figure 32: Standard Service Levels for server breakdowns and/or disk crashes. (von *Bülzingslöwen, 2006*)

Experiencing disaster events such as for instance combustions, earthquakes and/or airplane crashes, on the other hand, are divided into somewhat different categories at UBS and have a maximal downtime per disaster event instead:

- **Systemic –** is the most critical category, where the systems have national, international and external impacts. Thus the downtime cannot be more than three hours at most.
    - Subcategories of systemic are the so called **"transparent to market"** and **"<1h downtime",** where the system can only afford to break down for less than an hour.
- **Mission critical** – simply affect the UBS itself and is thus somewhat less critical than systemic. For theses categories of systems, the disaster recovery has to be executed within 24 hours.
- **Subsidiary** – is the last category and least critical with its upper limit of up to 72 hours, i.e. a breakdown is managed as long as it is below 72 hours.

(von Bülzingslöwen, 2006)

The recovery of disasters (so called **disaster recovery** - "Attributes of software that bear on the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it." *(Centre of Software Engineering (Essi-Scope), 2003, http://www.cse.dcu.ie/essiscope/sm2/9126ref.html)*) and general breakdowns can also be combined by for example stating that a Mission critical service has the Standard Service Level of "Premium +", which simply means that a mission critical service has to be handled within 24 hours, when the disaster has occurred, and that the service has to have an availability of 99.5% per year.

This example makes it obvious that some systems simply cannot be out of service and that risk analyses contribute to the overall categorizing and guidance of the expected level of performance. Thus systems have to be **reliable**, i.e. during a specific time period being capable of maintaining certain level of performance under given circumstances to support the business properly:

> "Eine Menge von Merkmalen, die sich auf der Fähigkeit der Software beziehen, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren." *(ISO/IEC 9126, 1994, p. 4)*

Systems are usually considered to be reliable, when a steady level, i.e. the level above the maximal downtime, of **availability** or the time that the system is up and running, can be assigned to the system. Measuring availability therefore includes the factors of time to failure and mean time to repair. Reliability on the other hand is only measured in terms of mean time to failure. (*Bass et.al., 1998*)

SOAs can guarantee a high level of reliability through the architecture per se, i.e. through e.g. its ESB or more precisely its ability of managing distributed components and systems through

the ESB. The distribution, reusability and componentization leads to a good chance of quickly regaining the original structure if disasters should occur. (*Elmasri & Navathe, 2004)* Furthermore, the ESB provides reliability through its good capability of assuring that the requests from requestors are transferred to providers, i.e. assuring the transportation of messaging. *(Bieberstein et. al., 2006)* In summary, reliability is achieved though a high and constant level of availability of architectural components and the communication between them.

BUSINESS PERSPECTIVE

QUALITY ATTRIBUTE (9): USABILITY

Having implemented a new architecture or simply rearranging an old architecture usually results in some sort of change for the users, due to changes in interfaces. As Löwgren (1993) states, most companies only see the profit that can be made with this new system, but to what extent the system will be fully used, is often not included in the calculation. When a company has chosen to implement a new or adjust the old architecture, it is simply assumed that users will know how to use every part of the new system and thus optimizes everything around the new investment.

Unfortunately, it is not always that easy. Surely it is likely that the invested fix costs will be covered within a short time due to the increased efficiency and thus save money, but the question is how this behavior evolves with time. If the employees do not manage to cope with the new interface, costs will soon start to exceed the savings. The first graph in *Figure 33* shows how the fixed costs (perfectly elastic) are covered (above the equilibrium) after a given amount of time. On behalf of the given example, it is assumed that the costs in the second graph are flexible and do not arise until after a certain time, i.e. when the business realizes that the users cannot use the interface. Furthermore, that the costs are presented with an inelastic curve is due to the assumption that as more time passes, costs are likely to increase faster. *(Gillespie, 1998)*

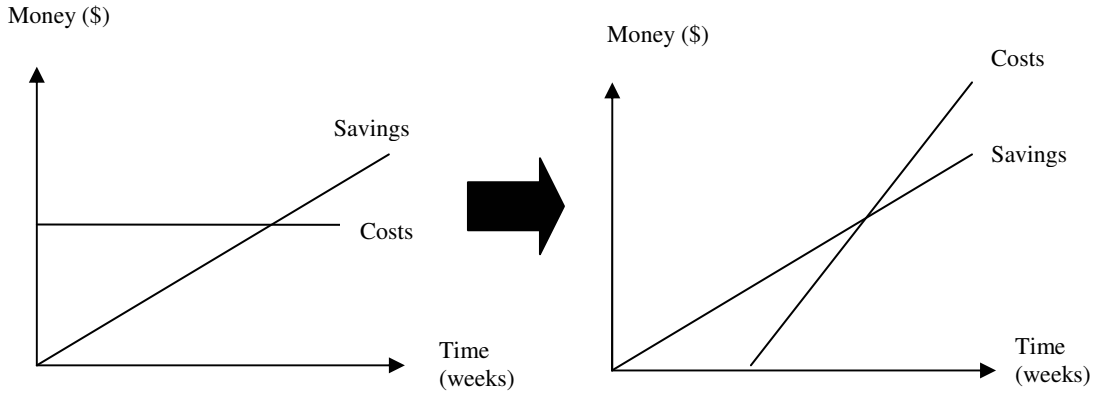


Figure 33: From having gained from the IT-Investment, the wrong treatment of usability may lead to exceeding costs. *(*Inspired by *Löwgren, 1993)*

To avoid risks of that kind, Bass et.al. (2003), ISO/IEC 9126 (1994) and Löwgren (1993) suggest the usage of the quality attribute usability, considering human factors as aesthetics and consistency in the user interface:

- **Relevance** – which points out that a system is only relevant as long as it serves the users' needs. *(Löwgren, 1993)*

- **Efficiency** – states how efficient a user does his/her work by using the system. Thus it is not the quality attribute efficiency, mentioned in section *Quality Attribute (6): Efficiency*. *(Löwgren, 1993)*

- **Attitude** – presents the users' emotional feeling towards the system, i.e. if he/she accepts the system. *(Löwgren, 1993)*

- **Learnability** – considers how easy it is for a user to learn to work with the system:

  > "Attributes of software that bear on the users' effort for learning its application." *(Centre of Software Engineering (Essi-Scope), 2003, http://www.cse.dcu.ie/essiscope/sm2/9126ref.html)*

- **Memorability** – represents the ability of users being able to remember the operations of the system over time:

  > "To what extent the user can remember how to do the system operations between uses of the system." *(Centre of Software Engineering (Essi-Scope), 2003, http://www.cse.dcu.ie/essiscope/sm2/9126ref.html)*

- **Understandability** – involves the understanding of the user, i.e. to what extent he/she really understands what he/she is doing:

  > "Attributes of software that bear on the users' effort for recognizing the logical concept and its applicability." *(Centre of Software Engineering (Essi-Scope), 2003, http://www.cse.dcu.ie/essiscope/sm2/9126ref.html)*

QUALITY ATTRIBUTE (10): BUSINESS FLEXIBILITY

Irrespective of market niche, most companies experience the pressure of other companies **competing** within their niche. This respect for substitutes, new entrants and the bargaining power of suppliers and customers drives companies to look for opportunities to gain more, regain or simply remain at a certain level of market share:

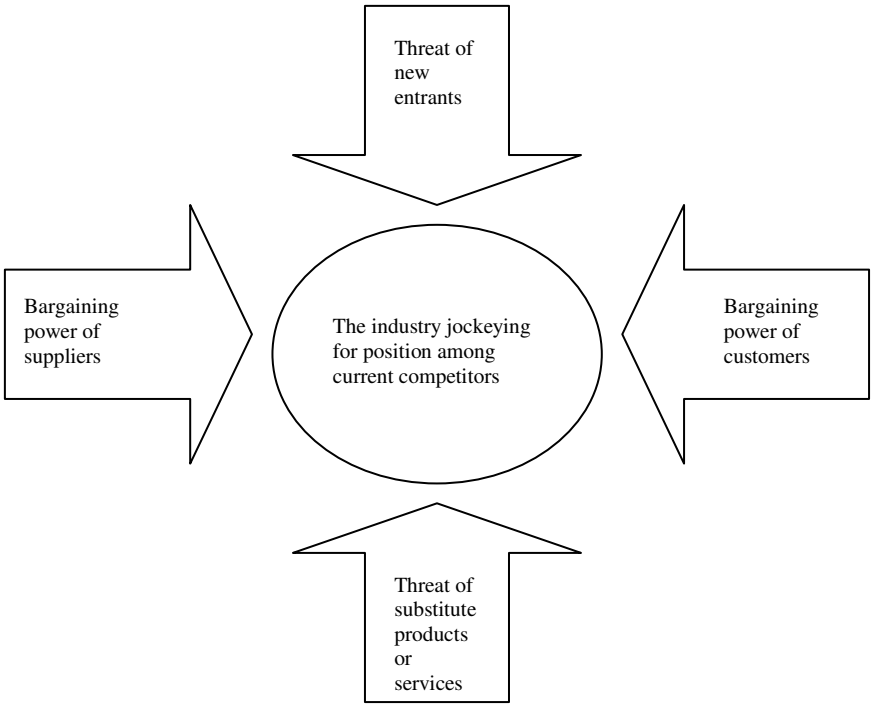


Figure 34: Porters five forces. *(Porter, 1998, p. 22)*

The greatest opportunity, suggested by Reinitz (2003, 2004, 2005), is to be able to provide flexibility, both towards customers and within the company. Flexibility in terms of meeting "[…] new market demands and to seize opportunities before they are lost or before the competition gets there first." *(Bieberstein et.al., 2006, p. 12*) Surely, for this the technical perspective, which amongst others is discussed in the quality attributes portability, maintainability, reusability and so on, is of great importance. After all, having an IT infrastructure that provides the users with the possibility of freely choosing the amount and kind of for processors and storage capacity that is used, leads to an incredible flexibility:

> "Kosten für die Hardwarenutzung fallen in Abhängigkeit von der tatsächlichen Nutzung der zugrunde liegenden Infrastruktur an, bspw. von Prozessor und Speicher." *(Krcmar, 2000, 2003, 2005, p. 144)*

However, one should not forget the more business oriented perspective, which involves looking at the "[…] **business operations** as a collection of interconnected functions […]" *(Bieberstein et.al., 2006, p. 12)*. To achieve flexibility within the business and indirectly gaining greater competitive advantages through reduced time-to-market, companies have to consider the outsourcing or streamlining of less important processes, as well as utilizing the advantage of service-oriented processes so that parts of the overall business process flow are delegated to different parts of the organization:

> "Through Service-Oriented Process, companies can delegate parts of their overall business process flows to different parts of the organization, each of which have direct and immediate control of the actual operation of the business." *(Schmelzer, 2005, http://www.zapthink.com/report.html?id=ZAPFLASH-20050127)*

Another business oriented perspective evolves around **product flexibility**, i.e. the "[…] degree of responsiveness (or adaptability) for any future change in a product design." *(Palani Rajan et. al., 2003, p. 1)*. As customers change their patterns of consumptions, the providers have to be able to adjust

their production in accordance. Having an IT architecture contributing to a flexible product design or redesign will on one hand reduce the costs of the products and on the other hand create the possibility of reducing the overall response time to consumers, "[…] by allowing quicker updates in the products and achieving higher levels of performance in a short span of time." *(Palani Rajan et. al., 2003, p. 1)*

In other words, the company per se in terms of more **efficient work streams**, **reduced costs**, **reduced development time** and **greater opportunities of quickly adopting to market changes**, and the customer, suppliers or other involved agents, through the increased chance of **higher quality of products and services** would all benefit from flexibility.

As an example, the flexibility difference between a silo-based architecture and SOA primarily lies between the vertical process executions and the more dynamically intertwined executions. Even though the silo-based architecture might be efficient, the ability to act flexible in accordance to business needs is lower than within the dynamic architecture. However, this only if the company has rearranged their business processes so that they are aligned to the new IT-architecture. *(Bieberstein et.al., 2006; Plummer, 2002; Maryoloy et.al., 2003)*

QUALITY ATTRIBUTE (11): DEVELOPMENT COSTS

As mentioned in the previous chapter, flexibility can be achieved by, for example, preferring a more service-oriented than silo-oriented architecture, as well as streamlining, adjusting or outsourcing processes in line with business demands and actions. All these aspects are covered in the development phase of a system and influence the costs of a system. As Krcmar (2002) shows with his life cycle model, the development phase is the most critical phase when it comes to costs and thus also quickly raises reasons for discussions:

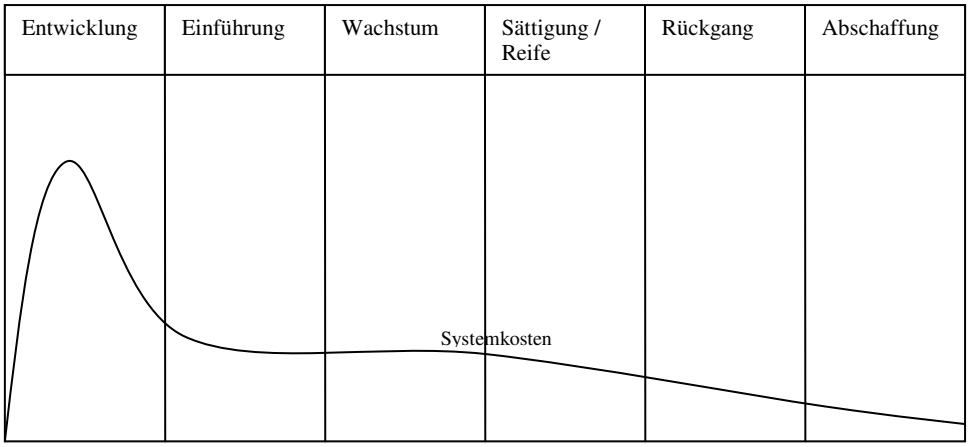


Figure 35: Life cycle of a system. *(Krcmar, 2000, 2003, 2005 , p. 146)*

To classify possible costs that occur during the development phase, Stoyan's (2004) model of fix costs for conception, content, implementation (design and IT), test and improvements (usability and IT) and project management provides a good basis:

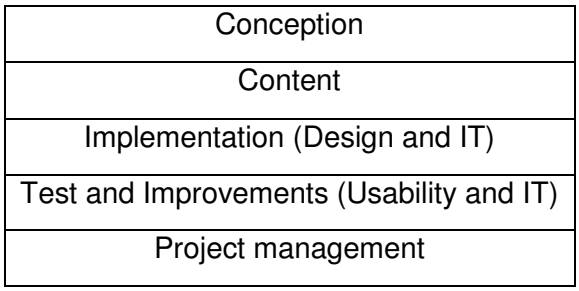


Figure 36: The biggest fix costs during the development phase of systems. *(Stoyan, 2004, p. 43)*

As can be noticed in *Figure 36*, the development not only involves technical aspects, but also the business processes from the previous chapter. In fact, these processes influence all five cost factors, with the emphasis on **conception, content gathering and project management**. Projects that work tightly with a business during the development phase, usually create matrix structures. If the company is structured and efficient in its processes, the project will have much better premises in succeeding, due to good cooperation between developers and customers. Thus project time, which is one cost factor, as well as direct project costs, can be kept low. *(Stoyan, 2004)*

For the **implementation** costs, reusability is one important SOA feature that leads to cost reductions (see section *Quality Attribute (3): Reusability*. More precisely, the costs of creating the components and/or services in the first step is higher than it was in traditional object-oriented programming for example, but having done this once and having access to an overall directory, leads to great future cost reductions.

> " [...] Business services can be encapsulated and abstracted in a way that makes them easy to utilize and assemble into component applications with minimal programming. Companies can utilize more skilled programmers for creating the underlying functionality and service definitions, which can then be reused by less technical programmers and visual application assembly tools." *(Gold-Bernstein, 2004, http://www.ebizq.net/hot_topics/esb/features/4894.html?page=2)*

Also, the feature of integrating loosely-coupled components and services reduces costs. More precisely, services that are loosely coupled can reduce the complexity of the architecture and hence reduce the costs of both integration and managing. The fact that SOA replaces "[…] multiple function calls at a fine level of granularity with coarser-grained, loosely coupled Services that can handle a wider range of interactions in a more flexible manner than API-based integration." *(Schmelzer, 2005, http://www.zapthink.com/report.html?id=ZAPFLASH-20050127)* reduces the overall cost of implementation.

Furthermore, already created components and services are only considered to be completely reusable, if they are created as "black-boxes" and thus already **tested and freed from potential bugs.**

> "Each service is like a black box that performs a specific function and has a published interface that accepts and defines inputs and produces defined outputs. Each service can be tested individually, then reused over and over. Interface testing is fairly straight forward, and can be automated using testing tools." *(Gold-Bernstein, 2004, http://www.ebizq.net/hot_topics/esb/features/4894.html?page=2)*

QUALITY ATTRIBUTE (12): RETURN ON INVESTMENT (ROI)

The introduction , has already shortly mentioned the fact that companies invest and expect some kind of value to the business in return. In fact, the amount invested - "[…] the purchase of new capital, such as equipment and factories […]" *(Gillespie, 1999, p. 52)* - within a company, is according to Gillespie (1998), directly depending on the expected return:

> "The level of investment depends on
> a) availability of finance
> b) interest rate
> c) the expected rates of return from the investment" (*Gillespie, 1999, p. 52*)

The expected return depends on factors, such as "[…] the initial cost of capital goods, expected costs, expected revenue and expected productivity" *(Gillespie, 1998, p. 52)*, which are all included in the traditional ROI formula on what kind of profit can be expected on a certain level of investment during a certain amount of time:

$$\text{Rate of ROI} = \frac{[(\text{Sales Price} - \text{Production Cost} - \text{Sales Cost}) \times (\text{Market Share} \times \text{Available Market} \times \text{Product Life} - \text{Development Cost})]}{(\text{Development Cost} \times \text{Time to Market})}$$

Figure 37: The traditional ROI formula. *(Green Hills Software Inc, 2006, http://www.ghs.com/MaximizeROI.html)*

Gunjan Samtani, the divisional vice president for IT at UBS PaineWebber, claims that such formulas are simply not usable for SOAs. *(Blakely, 2002)* Why is that? Due to the fact that SOAs have a more challenging architecture with the possibility of reusing assets, no specific features with readily identified returns are offered. Hence, one ROI or at least, a traditional one is not enough. *(Schmelzer, 2005)*

For the different formulated SOA ROIs, Samatani claims that one has to look beyond what the technology could possibly provide, i.e. "[…] the benefits must be weighed against risk factors that will impact the bottom line" (*Blakely, 2002, http://www.zdnet.com.au/insight/0,39023731,20270041,00.htm)*, and work iteratively during the composition. Hence, every time a new service is added to the SOA, corresponding ROI objectives should be defined for that particular service and weighted against the Web Service risk factors, such as "[…] quickly evolving technology, immature standards, insufficient support, quality of external Web Services, and security." (*Blakely, 2002, http://www.zdnet.com.au/insight/0,39023731,20270041,00.htm)*.

RESEARCH AREA – UBS

THIS CHAPTER IS NOT BASED ON THEORETICAL STATEMEMENTS, NEITHER IS IT AN EXPLANATION HAVING TO BE INCLUDED IN THE METHOD. MOREOVER, IT DOES NOT CONSIDER ANY EMPIRICAL RESULTS CONTRIBUTING TO THE OVERALL RESEARCH MODEL. HOWEVER, TO BE ABLE TO UNDERSTAND IN WHAT KIND OF BUSINESS THE RESEARCH MODEL WAS INTRODUCED AND WHAT THE ARCHIETCUTRAL CHANGE HAS IN FACT BEEN MEANING FOR UBS, THIS CHAPTER WAS ADDED.

UBS, as the leading bank of Switzerland and the worlds largest wealth manager with an invested assets of CHF 2652 billion *(UBS, 1998-2006e)* has after the merge in 1988 between the Schweizerische Bankgesellschaft (SBG; Union Bank of Switzerland, founded 1862) and the Schweizerischen Bankverein (SBV; Swiss Bank Corporation, founded 1872) its headquarters in Zürich and Basel. *(Known Library, 2004*) Its main functional areas are within wealth management business and business banking, investment banking, asset management and corporate and individual client banking *(UBS, 1998-2006a),* with a network of 1800 branches.

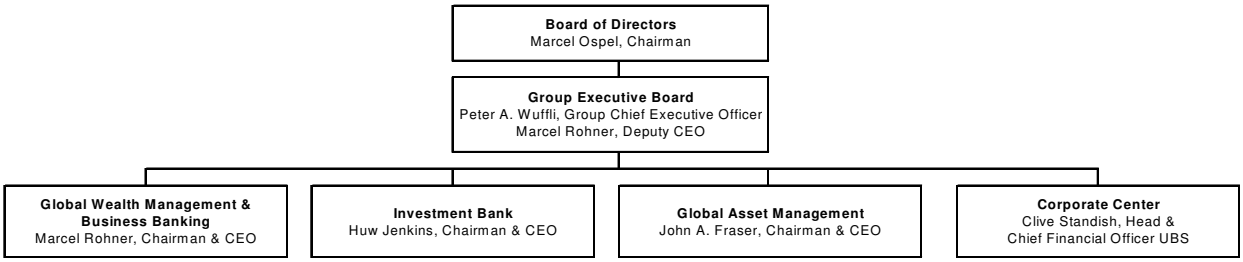


Figure 38: The organizational structure of UBS. *(UBS, 1998-2006b, http://www.ubs.com/1/e/about/ubs_group/group.html)*

At the current stage UBS is present in 50 countries all over the world, with more than 69,500 employees, of which 39% are situated in America, 37% in Switzerland, 16% in Europe and 8% in the Asia Pacific time zone. In Switzerland, UBS serves around 2.6 million individual clients and approximately 136,500 corporate clients, including "[…] institutional investors, public entities and foundations based in Switzerland" (*UBS, 1998-2006d http://www.ubs.com/1/e/about/our_businesses.html*).

The UBS report from 2005 shows a world wide profit of CHF 14,029 million, which is an improvement of 75% compared to 2004. Never has a Swiss corporation shown such a great return within one year. *(Tagesanzeiger, 2006)*

ABACUS, THE PREVIOUS IT ARCHITECTURE

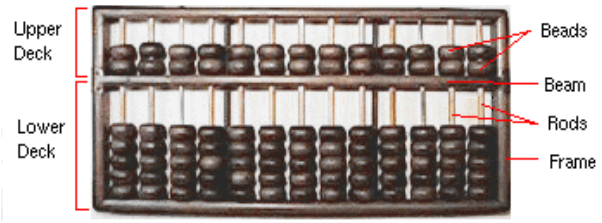


Figure 39: Abacus, a calculating tool. *(Unknown, 2003, http://www.ee.ryerson.ca/~elf/abacus/intro.html)*

The so called Abacus core banking system was planned and realized in the end of the 1970s and has been used up until now, i.e. for more than two decades. The IT system's general way of functioning can be symbolized in terms of an analogy of the old fashioned Abacus framework, i.e. with rods and beads that can be moved vertically up and down for the use of calculations *(Unknown, 2003, http://www.ee.ryerson.ca/~elf/abacus/intro.html).* In reality, there were no beads that were moved up and down on rods in the architecture, but the processes within the independently working silos, for each banking area, all relied on the same vertical Abacus principle[1].

To represent this a bit more clearly, let us consider the two different banking areas Mortgages and Fund accounts. In the Abacus architecture these two areas were represented in two different silos. Each silo, in turn, contained a function involving the entering of a new client, accepting his/her order, checking if the client is to mortgage or opening a fund account, calculating possible charges, charging interest and printing the confirmation for the client. Thus no reuse of equal or similar functions was applied. Furthermore, each function was vertically and sub-sequentially executed within the silo.

Some of the functions within the silos were also linked to data stores. These data stores in turn, were in almost all cases linked to one silo only, i.e. similar functions, in different silos, did not share a data store. The function "Accept order" within the Mortgage silo for instance, is linked to one specific data store. Likewise the "Accept order" within the silo Fund accounts is joined with a data store, but a different one than the one used from the Mortgage silo. Only the customer data was centralized in one data store, the so called Customer/Central Information File (CIF).

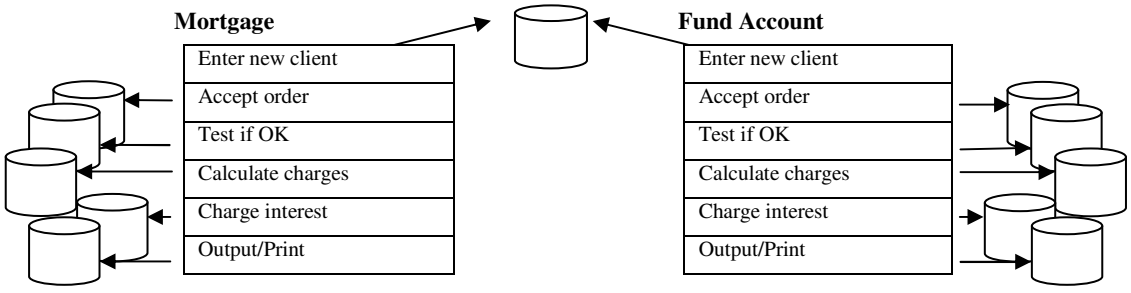


Figure 40: An example of the Abacus silos Mortgages and Fund accounts.

[1] The UBS did not consider the features of the calculating tool Abacus, when naming the architectural framework.

THE NEW ARCHITECTURE

During the end of the 20th century, UBS realized how difficult it was to update their current Abacus system according to business needs. Channels used separate data sources and applications to provide similar information, every silo worked on its own, no components were reused, a far too great amount of interfaces were present and developed solutions were only country based. (*Roik & Balzer, 2004)* Additionally, the demand from clients, employees and the bank increased rapidly. Thus, the main drivers for the initial system change were:

- "to increase flexibility and responsiveness to market conditions and customer requirements,
- to speed up time-to-market in developing new systems,
- to provide seamless support across multiple channels,
- to provide access on services and new combinations of services,
- to build applications based on modular, reusable components,
- to support enhanced communications and integration, and virtualization,
- to make better use of existing applications,
- to make inter-application communication and integration much faster, easier and less expensive,
- to use open standards,
- to reduce operating costs and asset intensity and complexity.
- to configure application from existing components rather than to develop applications from scratch" (*Roik & Balzer, 2004, p. 4)*

Considering this, thoughts around a new architecture arose quickly and the UBS – Global Wealth Management & Program Business Banking (Global WM&BB) initiated the so called Strategic Solution Program (SSP) *(UBS, 2005)*, with the aim to implement a new IT.

The first step towards the architecture, being partly implemented at the current stage, evolved around an OO-application. However, since the intentions were to move to a completely new architecture, the next and evolutionary step was taken shortly after the first one:

> "We spent some time developing a new application based on object orientation, but our real objective was to do the 'big thing' and move onto a new architecture - an architecture that would provide a clear view of the business services that we were running."
> *(Unknown, 2003, http://www.bankerme.com/bme/2003/jun/it_in_banking_2.asp)*

THE APPROACH

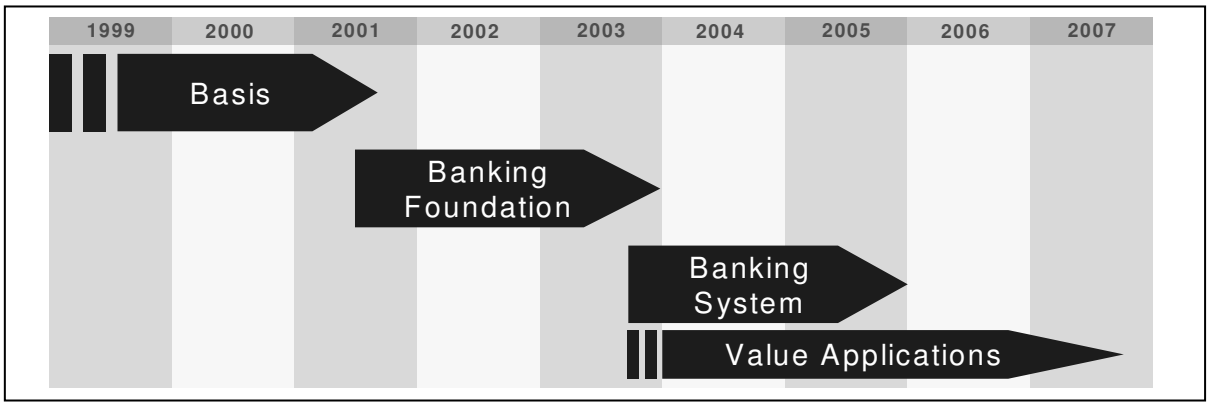


Figure 41: The process of the Abacus replacement. *(UBS Business & Application Architecture, 2004, p. 16)*

The main idea was to replace Abacus step-by-step, in close cooperation with the business, deliberately avoiding an overall and one-time replacement, due to the range and complexity of the system, as well as the enormous risk of possible unexpected errors and other incidents. The replacement is best presented in the following four phases:

- **The "basis" phase:**
  Between 1999 and 2001, the SSP focused on developing the overall project vision and objectives, as well as modeling and creating the new architecture and implementing data warehouse and logistic applications. This phase also included tests in real environments for usability and maturity of Web Services. *(Roik & Balzer, 2004; UBS Business & Application Architecture, 2004)*

- **The "banking foundation" phase:**
  The following phase lasted between 2001 and 2003 and involved the introduction of master data management, security and cash accounting, as well as financial and management accounting. More precisely, the defined methodologies from the preceding phase and the needed middleware "[…] were implemented to build the first heterogeneous and interoperable Web Services (December 2001 to June 2002)." (*Roik & Balzer, 2004, p. 18).* This middleware, also called BUS, leverages the possibility to offer multiple transparent services, with the same interface. *(Roik & Balzer, 2004; UBS Business & Application Architecture, 2004)*

- **The "banking system" phase:**
  Between 2003 and 2005, the "banking system" phase involved porting Abacus applications from Unisys to IBMs z/OS. *(UBS Business & Application Architecture, 2004)*

- **The "value applications" phase:**
  The currently ongoing phase with start in 2004 and expected end in 2007, is closely intertwined with the "banking foundation" phase and focuses on enhancing already defined functions to support possible business changes. *(UBS Business & Application Architecture, 2004)*

At the current stage, the Front-End part of the system is still under construction, while the Back-End is fully implemented.

ABACUS VS. THE NEW ARCHITECTURE

One of the obvious changes with the new architecture involves the platform change from Unisys to **z/OS** (Back-End services) and **DB2** (Business data). The other one is the fact that the silos have been exchanged to independently working, stateless and reusable business components or more precisely **business services** with interfaces. These **interfaces** are XML-based and are thus able to provide interaction between different kinds of business services, written in **Cobol** or **Java**, for example.

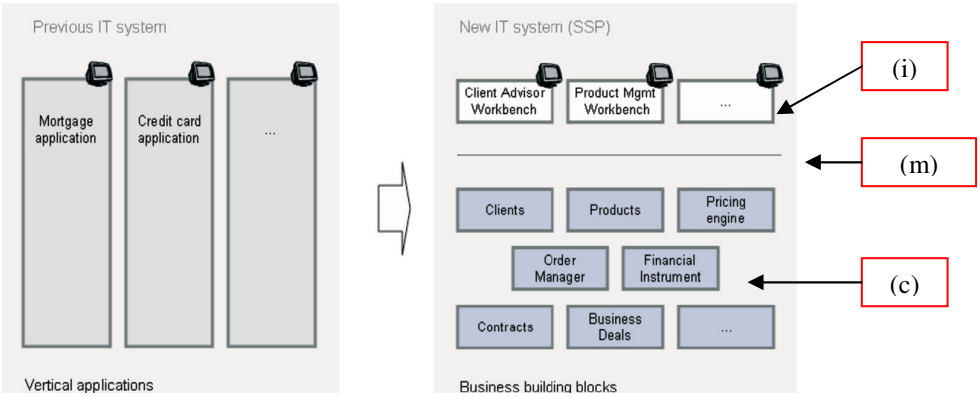


Figure 42: The old silo system and the new component (c) based architecture with interfaces (i) and a middleware (m). *(UBS, 2005, p. 8)*

Furthermore, the new architecture, as can be seen in *Figure 43*, provides a middleware, being a Multi-Channel Access Platform based on SUN Solaris servers (**MAP-OLU**[2]), and the Common Services Framework (**CSF**). The latter one is implemented with IBM's Message Queue (**MQ series**[3]) and **CICS**, the first one with WebSphere Application Server (**WAS**) and MQ. Through this middleware and with the help of the UBS specific middleware protocol **UBS XML**, which is based on the transport principle of MQ, business services can interact. The business services are found in the name service directories **i-SAC** and **Dyna/Rep**. (*Ebner, 2003*)

[2] OLU stands for Open Lan Unix, which is a UBS developed deployment package for Solaris servers, including also other standard packages used within the bank. *(Furth, 2006)*
[3] Today IBM MQ Series is called IBM WebSphere MQ, 14.07.2006

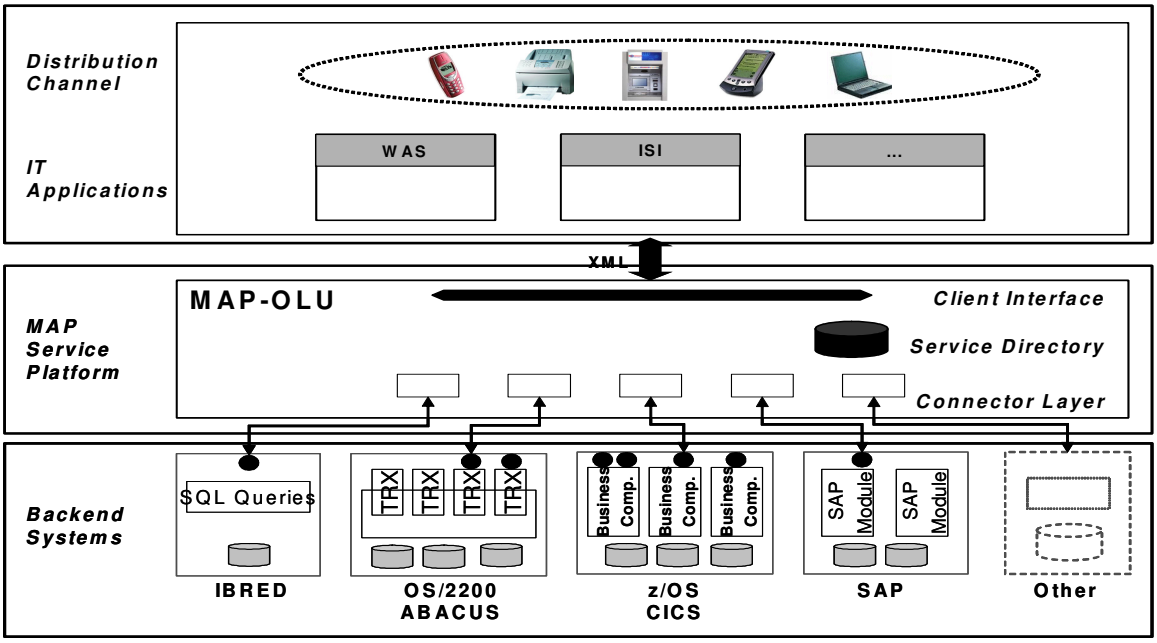


Figure 43: The new architecture of the UBS in terms of service-oriented concepts. (*Ebner, 2003, p. 51*)

Applying this new technology on the earlier mentioned banking area, i.e. 'Mortgages', the result will no longer include identical functions in different silos, but components that can be used in other banking areas, such as 'Fund accounts', as well.

To fully understand the new principle of the architecture, it is important to start with explaining the principle of the following three components:

- Partner
- Contract
- Product

All clients, partners, trustees or other people being involved in bank activities are in the new architecture defined as partners. Important is that a partner does not only symbolize one client etc., but could in fact present several. Banking services, such as bank account, maestro card etc., can be defined in terms of products. To be able to gain from one of these banking services, as partner, a contract has to be defined and signed. One product can appear in several contracts, but a contract can only present one product. *(Furth, 2006)*

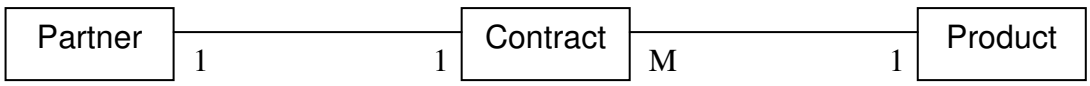


Figure 44: The relationships between partner contract and product. (Inspired by: *Furth, 2006*)

Now taking the mortgage and fund example again, this would simply mean that each partner is stored only once in the Customer Information (CI) database, no matter what he/she performs in combinations with the UBS. Having defined the partner and assuming that he/she wants to make an application of a mortgage, the next step is to add the mortgage contract to the concerned partner. In other words, the partner has to sign a specific contract to get his/her product. *(Furth, 2006)*

With the contract-process UBS will have an overview of all mortgages that are applied by a partner and also what mortgages can be related to a specific partner. In other words, in the

new architecture all individual partners no longer get flagged when applying and having a mortgage, for example, and no separate data store has to perform queries where the data in the CIF has to be combined with the flagging. Thus, the new architecture has already improved in terms of splitting up the silo structure, easing up the business process and combining data sources. *(Furth, 2006)*

Taking the example of calculating charges and interest for mortgages, also here a significant architectural change has occurred. As already pointed out, in the old architecture the interest and charges were calculated in a separate data store, having no relation to the actual mortgages. Due to the rearranging, the calculations are now done in one component or application, which has pointers at both the contract database and the Cash Core Account (ACC) database, where all mortgages are stored. *(Furth, 2006)*

In summary, one will realize that the former functions, 'Enter new client', 'Accept order', Test if OK', 'Calculate charges', 'Charge interest' and 'Output/Print', are no longer separated in different silos, but are combined in distributed and reusable components. Furthermore, all data stores, but CIF, are centralized and thus the overall amount, as well as redundancy, has been reduced.

IS THE NEW ARCHITECTURE OF UBS A SOA?

One might consider it strange never to find the term SOA explicitly mentioned in combination with the new architecture of the UBS. In fact, considering that the UBS started SSP already in 1999 and SOA, as a defined concept, did not enter the market until 2003, it would to some extent be incorrect to use the term in this context:

> "The state of the art in software will evolve markedly in 2003. Innovation and technical progress march on, even though the revenue growth of most software vendors was slowed by poor sales results in 2001 and 2002. Regardless of whether the rate of new IS development projects returns to the high activity levels of the late 1990s and 2000, new software technology will be brought to market, products will be repackaged and repositioned, and new standards — particularly Web Services — will begin to mature." (*Schulte, 2002, p. 1*)

Furthermore, the UBS does not use the classical SOA standards WSDL and SOAP, which of course also contributes to the difficulty in calling the architecture a SOA. However, as WSDL did not exist at all in 1999 and SOAP had just newly been introduced on the market, the UBS created standards of their own to achieve the features that today's SOA standards fulfill (see section *Enterprise Service Bus (ESB)*). Considering this and all the aspects mentioned in section *Abacus vs. the new Architecture* should definitely underline that this new architecture can be regarded as a SOA.

Figure 28 in section *SOA resume*, presented a visual view of SOA. A similar model is presented *Figure 45*. This version, however, is adapted to the concepts and architectural view of the UBS. Hence, comparing these two figures will indicate the architectural differences between the UBS architecture and a classical SOA.

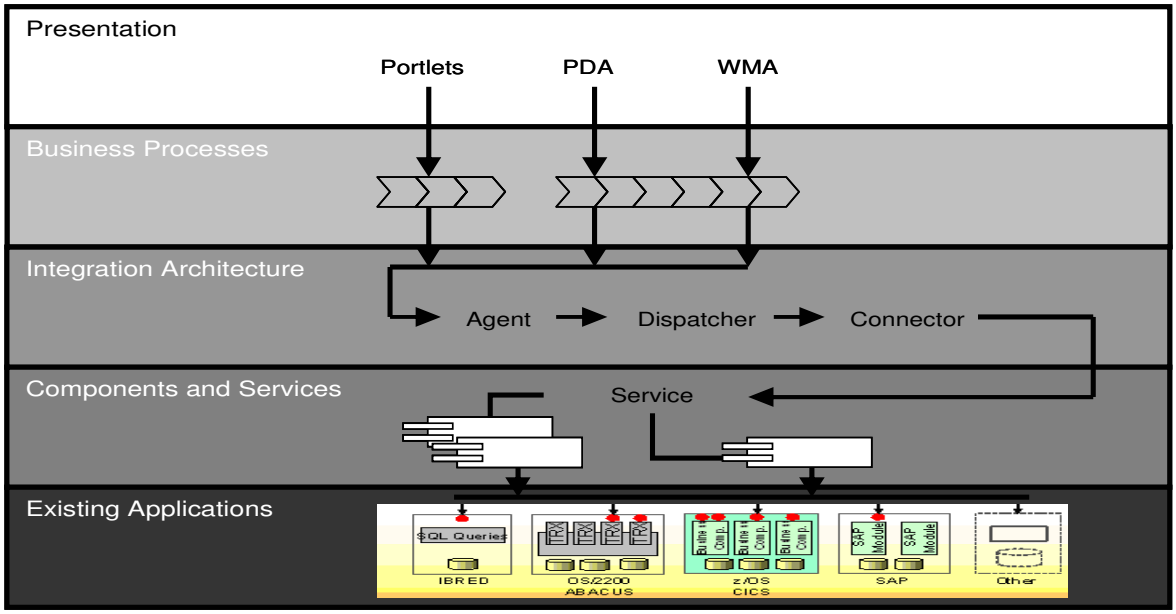


Figure 45: The service-oriented architecture of the UBS. (*Peisl; Ebner 2003*)

The greatest differences evolve around the following aspects:

- Additional mentioned channels, as for example the Personal Digital Assistant (PDA) and cash dispenser (WMA).
- Presenting the "Integration Architecture", with agent, dispatcher and connector horizontally instead of vertically and thus showing the close linkage between the "Business Process"-layer the "Components and Services"- layer:

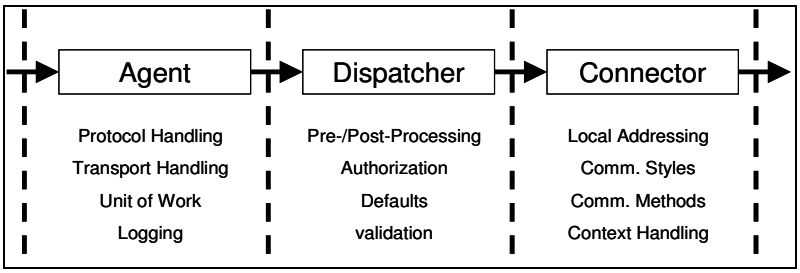


Figure 46: The UBS constructions of the integrations architecture. (*Ebner, 2003, p. 106*)

- A combined "Components and Services"- layer.
- Specific UBS Back-End Applications.

In summary, there are no great differences, it is more a question of how certain architectural features are presented.

RESULTS AND ANALYSIS

PRESENTING RESULTS AND ANALYSIS SEPARATE WAS CONSIDERED AS LESS APPROPRIATE. DUE TO THIS FACT, THIS CHAPTER WILL PROVIDE THE REDAER WITH BOTH RESULTS AND EVALUATIONS IN A CONSOLIDATED FORM. SHOULD THE READER BE INTERESTED IN RESULTS ONLY, THE ATTACHMENTS PROVIDE SUFFICIENT INFORMATION.

The method applied and theoretical foundation have up until now been discussed and presented in the chapters *Theoretical framework* and *Methodological approach*. The actual results from the overall research, however, are still to be presented.

To ease the understanding of this chapter, the results are presented both in terms of figures and written text. Furthermore, each result for the separate quality attributes is listed and discussed separately, to provide the reader with the same structure used in the *Theoretical framework*.

The analysis being conducted, due to the gathered results, is solemnly based on the following research question:

- **What impact has the newly implemented service-based architecture had on UBS?**

The research question focusing on to what extent the quality attributes can be regarded as SOA quality attributes, will be discussed below the sections of *Technical perspective* and *Business perspective*, as these two perspectives combine all mentioned attributes:

- **What quality attributes can be regarded as SOA quality attributes?**

The remaining two research questions are discussed at the end of this chapter, as these are best answered on behalf of the discussions being held in combination with each attribute:

- **Is the SOA Quality Evaluation Model applicable to SOA implementations?**
- **Does the interaction between the quality attributes affect the overall outcome of the model?**

TECHNICAL PERSPECTIVE

That the eight technical attributes can be regarded as quality attributes can be derived from the quality definitions being given in chapter *Quality*. After all, six (modifiability, reusability, integrability, security, scalability and reliability) of these eight attributes are measurable (see the theoretical definitions of the attributes) and were defined by UBS as vital for success (Likert scale value = 4). As measurable success factors were regarded as being contributors to increased quality (see chapter *Quality)* the conclusion to be drawn simply has to evolve around the fact that these six attributes are essential when achieving a certain level of quality. Also the remaining two attributes portability and efficiency are potential quality attributes, as they also are measurable and being considered important by UBS (Likert scale value = 3).

The fact that UBS received questions asking to what extent a certain attribute is vital or critical for their success, indicates that the quality being discussed is in fact business based. However, if the questions would have asked whether or not an attribute is vital or critical for the IT architecture, the overall picture of the benefits or limitations of the new architecture would have vanished. Besides that an IT architecture is implemented to support its business and therefore the criticality out of the business perspective is of greatest interest.

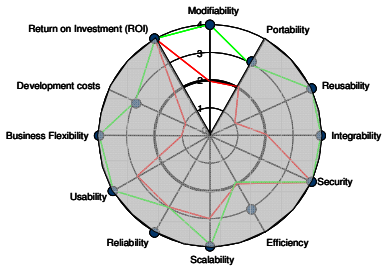
The question now remaining is whether or not these quality attributes can be considered as SOA quality attributes. Once again this conclusion is highly UBS related, due to the research being conducted, but as stated in the chapter *Methodological approach*, at this point in time no other bank in Switzerland has implemented a SOA. Hence, currently the UBS answers have to be considered as general statements for Switzerland. Due to this, the five attributes modifiability, reusability, integrability, security and scalability all can be regarded as SOA quality attributes (Likert scale value = 4), as all five were said to be strongly present in the implemented SOA. Portability and reliability are also closely related to SOAs (Likert scale value = 3). Efficiency is according to UBS the least appropriate (Likert scale value = < 2).

In summary:

Description	Likert scale value	Attributes
Obvious quality attributes:	4 (weighting value)	modifiability, reusability, integrability, security, scalability and reliability
Potential quality attributes:	3 (weighting value)	portability, efficiency
Obvious SOA quality attributes:	4 (‘As-it-is’ value)	modifiability, reusability, integrability, security and scalability
Potential SOA quality attributes:	3 (‘As-it-is’ value)	portability, reliability
Less appropriate SOA quality attribute:	=< 2 (‘As-it-is’ value)	efficiency

Figure 47: The obvious, potential and less appropriate technical SOA quality attributes.
(Attachment 6: UBS answers)

QUALITY ATTRIBUTE (1): MODIFIABILITY



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
4	2	4

Figure 48: The answers for the quality attribute modifiability (*Attachment 6: UBS answers*)

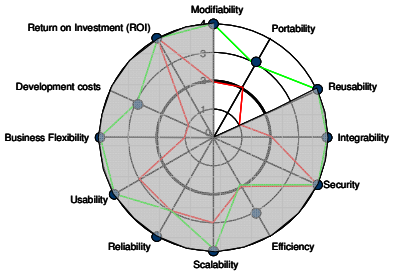
According to the received results, modifiability has been stated as one of the 10 most important quality attributes at UBS, with the 'Yes, I strongly agree' answer on the question: 'Is one of the companies target goals to be able to modify attributes within the company, without increasing the complexity and rearranging the structure of the architecture?' (see *Attachment 5: Questionnaire - SOA quality attributes*). With the chosen answer UBS points out the essence of being able to change, for example, processes, technologies, components, services, rules and other attributes being architecture related, without having to rearranging the implemented architecture. (*Quality Attribute (1): Modifiability*). Moreover, the chosen answer states (see *Attachment 5: Questionnaire - SOA quality attributes*) that having this kind of architecture contributes to the overall success of the company, i.e. in accordance with the discussion held in the *Theoretical framework*, having an architecture not supporting modifiability can be considered as critical for business success.

Having stated this, it becomes obvious why UBS started SSP and implemented the new service-oriented architecture. After all, SOAs make modifiability profitable through its modularization, encapsulation, loose coupling of components and configurable applications. Thus, as mentioned in the section *Quality Attribute (1): Modifiability*, UBS will now be able to respond quickly to actions such as deleting unwanted capabilities, restructuring, adapting to new operating environments and extending or adding attributes by simply configuring and managing the new architecture. This, in turn, has been stated by the answer 'The architecture supports business agility and has an open structure.', i.e. a 'Yes, I strongly agree' on the 'As-it-is' question.

The interesting aspect of the three received answers is the 'The question cannot be answered.' response on the question, alluded on the old architecture, 'Is the architecture open for attribute modifications, as business situations change?'. This means that the respondent either does not have the knowledge about the question asked or simply that Abacus was never discussed in terms of modifiability. No matter what, both alternatives provide a neutral answer, showing nothing of how modifiable the old architecture was.

In summary, the quality attribute modifiability is highly essential for the UBS as a business and therefore has been strongly considered when choosing the new architecture. The new architecture, with its open structure, is most likely by far more modifiable, considering that Abacus was a silo-based. This, however, was not confirmed by UBS-respondents.

QUALITY ATTRIBUTE (2): PORTABILITY



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
3	2	3

Figure 49: The answers for the quality attribute portability (*Attachment 6: UBS answers*)

Having a portable architecture, i.e. the opportunity of adapting to different environments (e.g. platforms, countries) and considering cultural internationalization (e.g. language and legislative adjustments and/or specific system preferences) without having to make major architectural changes, is regarded as important, but not vital at UBS. In other words, by giving a 'Yes, I agree' on the question 'Does the company see portabilities, such as being able to update the IT-architecture with the latest hardware and/or to transfer the architecture to other areas, as vital?', the quality attribute is regarded as one of the important quality attributes, but on the other hand not as one of the essential ones out of a business perspective. The added remark 'Cultural Internationalization is very important; platform independence is less important' from respondents clarifies why the quality attribute was not marked with a 'Yes, I strongly agree'.

That portability was of particular importance before the new architecture was implemented can partly be derived form the answer being given on the question 'Is the architecture suitable for adapting to new environments?'. After all, the 'I neither agree nor disagree' may point out that UBS had not been regarding this aspect when implementing Abacus in the first place. Furthermore, after having implemented the old architecture, there was no real opportunity to achieve portability. On the other hand, this answer might simply state that the respondents did not possess any knowledge about to what extent Abacus was portable or not. A third reason why this answer was chosen might be due to the fact that all other answer alternatives were not regarded as applicable. On the other hand, are the other answer alternatives quite broadly formulated, covering both portability and non-portability, hence the given alternatives should provide the respondents the needed choices.

Assuming a lack of portability consideration in the old architecture, it seems as if at least in the new service-oriented architecture this quality level has gained more interest. With the answer 'Yes, I agree' on the question 'Is the architecture suitable for adapting to new environments?', the alternative 'The architecture is relatively portable and supports limited platform independency. Cultural internationalization, however, is not supported.' UBS shows that the aim of achieving and most of all considering portability during SSP has been fulfilled.

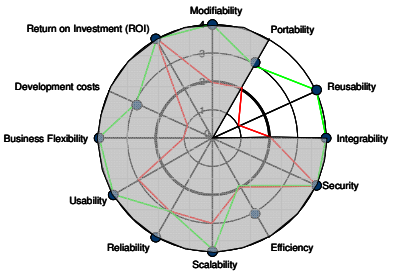
What might seem somewhat strange is the circumstance that the portability is only regarded as limited and that cultural internationalization is not supported at all. After all, SOAs per se are to provide a high level of portability with its information hiding principle, as mentioned in section *Quality Attribute (2): Portability*. Furthermore, UBS has the intention of implementing the architecture in other countries around the globe and by stating that the architecture does not support cultural internationalization simply is not in accordance with the aim set up for the architecture (see remark mentioned earlier in this chapter).

What might have forced the respondents to this answer could be the fact that the answer alternative regards two aspects, i.e. platform independency and cultural internationalization, and both of them are not equally important for UBS. However, the second question is not discussing the importance of the attribute in any of the quality attributes, but the presence of the attribute within the architecture. Thus, as SOAs are said to have a high level of portability, i.e. both platform independent and suitable for cultural internationalization, the answer alternative 'Yes, I strongly agree' would have been the applicable.

However, it must be admitted that the answer alternative for 'Yes, I agree' should instead have been formulated as follows: 'The architecture is relatively portable and supports limited platform independency and cultural internationalization.'

In summary, the ability of making platform changes and to even move the architecture abroad is important for UBS, but still this is according to the given answer not completely achieved with the new service-oriented implementation. However, it seems as if the new architecture at least is more focused on portability than the old one, where this quality level obviously was not considered. Furthermore, the answer given might not have been fully the intended one.

QUALITY ATTRIBUTE (3): REUSABILITY



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
4	1	4

Figure 50: The answers for the quality attribute reusability (*Attachment 6: UBS answers*)

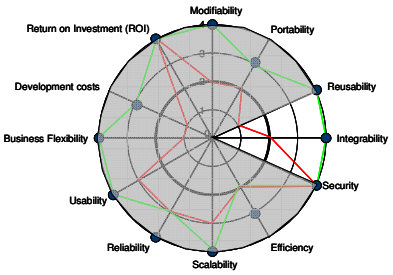
As stated by the respondents, UBS sees reusability of already available resources as a vital process and critical for success. Thus, architectural components, for example, should constantly be reused when using similar functions/features in other parts of the system or other systems. In other words, reusability is another of the 10 most important quality attributes. This is not too surprising considering that modifications in strongly coupled architectures, as mentioned in the section *Quality Attribute (3): Reusability,* will turn out to be far too inefficient. Loosely coupled components, on the other hand, can improve efficiency, since they are possible to reuse and thus lower, amongst others, development and maintenance costs.

Reflecting over the answer of how important reusability is for UBS, the 'Yes, I strongly agree' on the 'As-it-is' question 'Does the architecture support the reusing of already existing structures and components?" seems natural. After all, UBS would not go from an architecture, not providing reusability ('As-it-was' question) – 'Similar components and structures tend to occur in the architecture, due to lack of reusability.' – to another one having this deficit. This especially if reusability is considered to be a CSF.

Being aware of the fact that SOAs are based upon reusable components, it possible to quickly deduce that the new architecture at UBS is more in the direction of a SOA than, for example, a silo-oriented architecture. Surely, this has already been stated in section *The new architecture*, but the positive aspect is that the answers ('As-it-is" question) – 'In the architecture a function that is the same in one or more business areas is only implemented once in the whole company.' – for this quality attribute confirms this definitely.

In summary, UBS has managed to achieve the desired level of reusability within its architecture by implementing the new architecture.

QUALITY ATTRIBUTE (4): INTEGRABILITY



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
4	2	4

Figure 51: The answers for the quality attribute integrability (*Attachment 6: UBS answers*)

Considering that answers for other attributes have shown proof of, for example, UBS having the ability of easier making modifications due to loosely coupling and applying reusability due to a more component or service oriented architecture, it is not too surprising that also integrability is regarded as being strongly presentable in the new architecture. Having stated that 'The new architecture integrates its components/services through interfaces over a middleware with protocols.' once again confirms that the new architecture has the features of a SOA.

The importance of maintaining such an architecture, out of a business perspective, is also in this attribute stated through the chosen answer - 'Without a properly working integration between architectural components, the business will not even be able to perform the easiest business processes' on the weighting question 'Is it vital for the ongoing business that parts of the architecture can interact with other parts via well defined interfaces?'. Thus, UBS is not able to perform the easiest business process without having a properly working integrability.

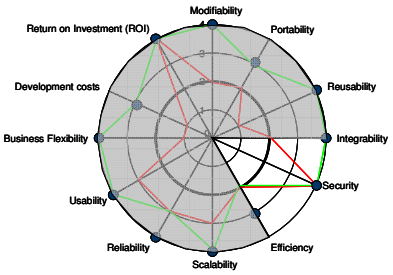
The neutral answer 'I neither agree not disagree' on the 'As-it-was' question 'Does the architecture provide interaction between architectural parts through an interface language, a middleware (e.g. ESB), a directory, protocols and/or interfaces?', might once again be an example of the respondent's non-familiarity with the quality attribute or because the respondent did not read the other disagreeing alternatives carefully enough. After all, considering that the old Abacus architecture was silo-based, at least on of the following two answers should have bee applicable:

- "The architecture does not support integrability between components and services."
- "The architecture does not need integration capabilities."

As a worst case scenario, this answer alternative was chosen since all other alternatives simply were not applicable to the old architecture, i.e. the architecture does not have the classical silo features.

In summary, integrability is seen as a CSF and is therefore also highly present in the new architecture. To what extent the old architecture used integrability is not identifiable on behalf of the received answers.

QUALITY ATTRIBUTE (5): SECURITY



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
4	4	4

Figure 52: The answers for the quality attribute security (*Attachment 6: UBS answers*)

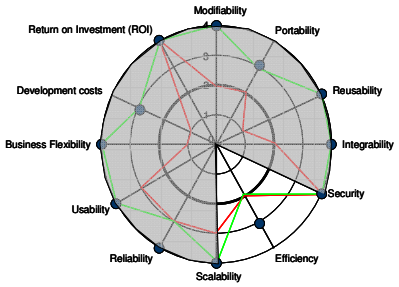
Section *Quality Attribute (5): Security* mentioned that security aspects become especially important due to the introduction of Web Services, but at the same time it is mentioned that authentication, authorization and access control, as well as encryption are just as important without Web Services. This statement is confirmed by the given answers, i.e. a 'Yes, I strongly agree' for all questions. In other words, no matter what kind of architecture is used, UBS prioritizes security and classifies the quality attribute as a CSF – 'Functional security is vital for the business. (Security is a CSF.)'

What might seem somewhat strange are the answers on the 'As-is-was" and 'As-it-is' question 'Does the architecture include authentication checks, authorization, access controls and/or provide firewalls between networks and use digital signatures for the exchange of messages?'. In fact, it was assumed that the respondents would chose the alternative stating that the architecture had a good security, but without Web Service and thus network protection, i.e. 'The architecture provides security, but not on a web basis, i.e. authentication checks, authorization and access controls take place, but not network protections.' Instead the respondents gave the following answer for both architectures: 'The architecture provides the involved parties with very good security, including network protection."

As it seems, either the structure of the old architecture included aspects not having been published and discovered during the research or simply the formulation of the answer alternative 'Yes, I strongly agree" was to little focused on network protection for Web Services.

In summary, security is the only quality attribute with ROI that shows the highest possible level of achieved quality both in the old and new architecture. This is also supported by the business, which sees this attribute as vital.

QUALITY ATTRIBUTE (6): EFFICIENCY



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
3	2	2

Figure 53: The answers for the quality attribute efficiency (*Attachment 6: UBS answers*)

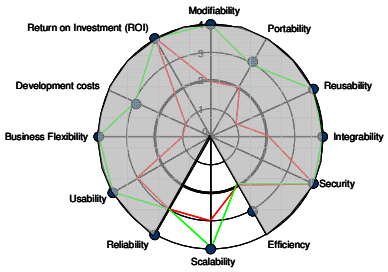
The ability of achieving a suitable performance in terms of IT architecture used, as discussed in section *Quality Attribute (6): Efficiency*, and amount of resources, is seen as important at UBS, but not vital, for business agility. In other words, the IT efficiency is a factor that influences the overall business operability and hence architecture and resources being used have to maintain a relatively high standard to achieve satisfaction.

This satisfaction, however, was obviously not achievable or measurable in neither Abacus nor in the new architecture if one considers the answers being given on the question 'Does the architecture act efficiently with the available amount of hardware, business processes and generated performance?'. After all, the respondents settled on the answer 'I neither agree nor disagree', i.e. 'The question cannot be answered.'. The exact reason for the chosen answer is unfortunately not possible to evaluate, but most likely the respondents did not have the needed efficiency numbers to give a specific answer.

That a different answer was expected is mainly due to the theoretical aspects being discussed in section *Quality Attribute (6): Efficiency*. There it was said that efficiency is greatly influenced by the architectural structure being applied, i.e. the IT being present, the interaction between different architectural parts, the process synchronization, the queue size of requests, the amount of latency, as well as the structure of business processes. These factors, are in general said to improve with a SOA. Hence, changing from a silo/-based architecture to a SOA should generate some kind of or even significant change in efficiency.

In summary, efficiency is regarded as important at UBS, but to what extent the old or new architecture is actually efficient is not identifiable.

QUALITY ATTRIBUTE (7): SCALABILITY



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
4	3	4

Figure 54: The answers for the quality attribute scalability (*Attachment 6: UBS answers*)

Once again the respondents defined a quality attribute as being vital for the success of the company. More precisely, UBS stated that the ability to increase throughput under an increased load of resources is even critical for the success of the company – 'Scalability is critical for the success of the company. (Scalability is a CSF.)' and should therefore be presented in the available architecture.

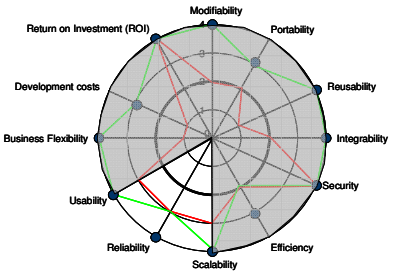
That scalability was considered essential in the old architecture already, is confirmed by the answer 'Yes, I agree' on 'As-it-was' question – 'The architectures provides an increase, but not proportional, in performance, when adding more load. Thus the architecture maintains scalability features.'. As it seems, Abacus was able to support UBS with the ability of maintaining an efficient level of throughput under an expansion, even though the increase was not proportional.

With the new architecture UBS has managed to improve the minor scalability shortage of Abacus and support their actions towards business success, i.e. the new architecture provides a proportional increase in performance - 'The architectures provides a proportional increase in performance, with adding of more load. Thus the architecture maintains excellent scalability features.'. With this improvement UBS has good preconditions for business agility, i.e. being agile to new areas and partly also unexpected situations.

A reason why scalability has improved through the architectural improvement might, amongst others, be due to the middleware being used in the new SOA. Heineman & Councill (2001), for example, stated that scalability is best achieved through a middleware. As a middleware is implemented at UBS with the new architecture (see section *The new architecture*), this might be the decisive point or at least contributing one why this amount of scalability has increased to this extent.

In summary, scalability, a vital success factor for UBS, improved as the new service-oriented architecture was implemented. With the silo-based architecture, UBS could perform quite good business and architectural expansions.

QUALITY ATTRIBUTE (8): RELIABILITY



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
4	3	3

Figure 55: The answers for the quality attribute reliability (*Attachment 6: UBS answers*)

As already stated in section *Quality Attribute (8): Reliability*, UBS maintains a principle where different kinds of systems that experience breakdowns are specified with Standard Service Levels and maximal downtimes. By doing so, UBS classifies its systems, which ones cause greater risk for the company than others. A system, having the 'Premium +' Standard Service Level, can, for example, only be out of service during two hours per event. A similar principle is used for potential disasters as combustions, earthquakes and/or airplane crashes. This well developed risk management is typical for companies having a high focus on reliability. In other words, the answer 'Reliability is vital for the survival of the business. (Reliability is a CSF.)', is not really surprising.

The answer 'The architecture is relatively reliable, since safe messaging/packaging (interaction between components) is assured.' on the question 'Does the architecture support systems with high reliability?', however, might be somewhat more surprising. Not surprising in the sense, that the both architectures are not completely available and reliable, i.e. 'The systems in the architecture all keep a high level of reliability, due to high availability of architectural components and good interaction between these components.', but rather that this answer was valid for both the old and new architecture. On the other hand, this does only show that good reliability can be achieved no matter what structure is being used. This, even though SOAs are said to be more reliable through its distributed service orientation, and through the ESB (see section *Quality Attribute (8): Reliability)*, i.e. the chance of quickly regaining the original structure if disasters should occur by using SOA typical distribution, reusability and componentization.

That not all systems maintain a high level of reliability, for example, might be due to a lacking overall IT-strategy, where it is usually defined how newly implemented systems should be treated and registered. In other words, even though the architecture at UBS is open and suitable for the adding of new systems, this has to be correctly treated, to achieve a good reliability. Strangely enough this problem should, in fact, have been more severe in the old architecture, since added systems were possible to be implemented without interaction to other systems. In other words, systems, without proper registration, being implemented in different business areas in a silo-architecture should be more difficult to detect than in a SOA, where interfaces show what services or components are bundled at the end-points.

In summary, UBS has a high focus on reliability, which also has been implemented in the architecture, old as new.

BUSINESS PERSPECTIVE

Applying the same assumptions being presented in the discussion about the technical attributes, i.e. quality, measurability and essentiality for success (see *Technical perspective*), three (usability, business flexibility, return on investment (ROI)) out of four possible business attributes are to be regarded as quality attributes. The development costs are not vital for UBS, but can still be regarded as important. Hence, this attribute is still within the range, i.e. above two on the Likert scale, of being a potential quality attribute.

The SOA applicability of these business attributes, are just as in the *Technical perspective* based on the UBS 'As-it-is' answers. Considering this, the attributes that were seen as obvious quality attributes also represent obvious SOA quality attributes. When it comes to the development costs, these remain at their potential position also in combination with SOA applicability.

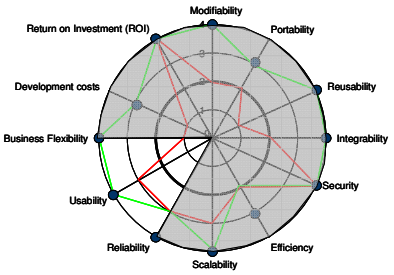
As can be concluded, in comparison to the technical attributes in the section *Technical perspective*, none of the business attributes were by UBS stated as less appropriate, neither as simple quality attributes nor as SOA quality attributes.

In summary:

Description	Likert scale value	Attributes
Obvious quality attributes:	4 (weighting value)	usability, business flexibility, return on investment (ROI)
Potential quality attributes:	3 (weighting value)	development costs
Obvious SOA quality attributes:	4 ('As-it-is' value)	usability, business flexibility, return on investment (ROI)
Potential SOA quality attributes:	3 ('As-it-is' value)	development costs

Figure 56: The obvious, potential and less appropriate technical SOA quality attributes.
(Attachment 6: UBS answers)

QUALITY ATTRIBUTE (9): USABILITY



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
4	3	4

Figure 57: The answers for the quality attribute usability (*Attachment 6: UBS answers*)

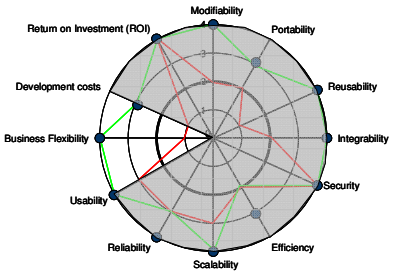
At current stage UBS has not implemented the Front-End of the overall architecture, which of course makes it somewhat difficult to specify what kind of quality level is currently obtainable with the new architecture. Thus, the answer 'Yes, I strongly agree' on the question 'Are users of the systems generally speaking satisfied with the ease of use of the user interface?', out of a 'As-it-is' perspective, is in fact confusing. What can be assumed is that the respondent has chosen this answer, since the Front-End of the new architecture is aimed at supporting the user in its actions and to provide learnability, understandability, as well as memorizeability – 'The architecture supports the development of systems that are easy to learn, understand and memorize. Furthermore the system serves the users needs.'.

However, assuming this before the user has even hade the chance of being involved in the Front-End must be regarded as a critical issue. After all, if it assumed that the user will have optimal opportunities of interacting with a system, no more effort will be invested to really achieve this high level of quality. As a reaction, UBS might have to experience exceeding costs, since the users will not be able to generate the expected amount of output of the system. On the other hand, is this perhaps not the most critical issue for UBS, since the whole SSP has been based on a very generous budget, where the focus has been less on profit than on the overall expected outcome.

The answer being given to describe the system interfaces of Abacus indicates great satisfaction – 'The architecture supports systems that serve the users needs or are easy to learn, understand or memorize'. Considering that the far too great amount of interfaces was a reason for wanting to replace the architectures in the first place (see section *The new architecture*), makes this answer somewhat hard to understand. After all, a great amount of different interfaces tends to make the user more reluctant to understand and memorize the actions required. Disregarding this perspective, both the old and new architecture supports on of the success strategy of UBS, i.e. to maintain satisfied and challenged systems users – 'Satisfied and challenged system users is vital for the company culture. (Usability is a CSF.)'.

In summary, UBS sees usability as a vital attribute and assumes to achieve an even higher level in the new architecture, in comparison to the old.

QUALITY ATTRIBUTE (10): BUSINESS FLEXIBILITY



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
4	1	4

Figure 58: The answers for the quality attribute business flexibility (*Attachment 6: UBS answers*)

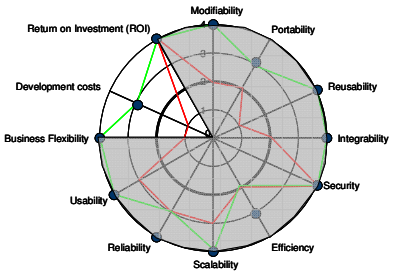
As competition usually is regarded as a critical factor for companies, (mentioned in section *Quality Attribute (10): Business Flexibility*), both technical and business-oriented flexibility gain more and more in importance. The importance of the technical flexibility was confirmed within the quality attributes in the recent sections, the business flexibility, however, has not specifically been confirmed yet. However, with the answer 'Yes, I strongly agree', i.e. 'Flexibility of business processes, products and IT is critical for the success of the company. (Flexibility is a CSF.)', also the assurance of the essence of business flexibility is given.

According to the received results the possibility of quickly adjusting to new market demands, due to optimized business processes and products, seems to have changed remarkably with the implementations of the new architecture. The question 'Are the business processes and products in the architecture flexible enough to adjust quickly to new market demands?' was, for example, answered with a 'No, I disagree' for the old architecture and a 'Yes, I strongly agree' for the new architecture. In other words, the new architecture provides extraordinary opportunities for gaining competitive advantages, while Abacus did not.

Applying this to the fact that the new service-based architecture is dynamically structured, the ability of rearranging and optimizing business processes surely also has increased the ability of producing products more efficiently. With this new architecture, the UBS no longer has to apply the vertical and sub sequential executing processes and thus decreases time and money for producing a product. On the contrary, it is likely that UBS has outsourced or even streamlined some of its processes.

In summary, UBS once again defines a quality attribute as vital for business success and has managed to achieve a high level of quality with the new architecture, which Abacus had not provided.

QUALITY ATTRIBUTE (11): DEVELOPMENT COSTS



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
3	1	3

Figure 59: The answers for the quality attribute development costs (*Attachment 6: UBS answers*)

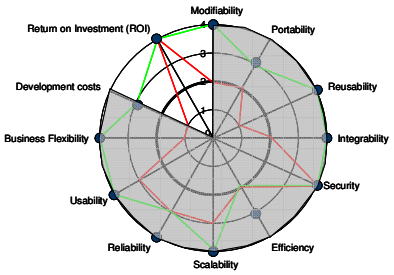
Keeping the costs, being generated through the development of a new system, low, is important according to the respondents, but not vital and neither part of the overall IT-strategy of UBS. Instead it is said that 'Having an architecture with reusable components and optimized business processes is important for the business.'. A statement like this indicates that UBS is not particularly interested in optimizing the system costs to its limit, but rather keep them realistically low. It therefore also of interest to maintain an architecture where the combination of standardization and self-development features is optimized.

The response 'Yes, I agree', i.e. 'The architecture provides the business with the possibility to implement new systems to relatively low costs.', on the question 'Can costs be kept low due to optimized business processes and reusable IT components, when new systems are developed within architecture?' simply underlines the answer being given in combination with the quality attribute reusability. In other words, by using a service-oriented architecture, UBS has the possibility to introduce new systems at a lower cost than with the old silo-based architecture. Thus, the reusability of services to create a new system generates the less costs than if the system was made form scratch.

Even though the implementation per se is of great importance for UBS when reducing costs, also the interest of optimized business processes was mentioned. As discussed in section *Quality Attribute (10): Business Flexibility*, together with a service-oriented architecture also the business processes are adjusted, i.e. either they are outsourced or streamlined. This close correlation between business process and the architecture did in contrast to the new architecture not exist in the old architecture – 'The architecture does not provide cost efficiency, i.e. low costs might have been kept, but not due to reusability and efficient processes.'.

In summary, UBS regards low development costs of systems as important, but not vital for the success of the business. This statement has most likely been some sort of aspect when choosing the new architecture, even though it was not the deciding point. At least, the new architecture provides the possibility of reducing development costs, in contrast to the old one.

QUALITY ATTRIBUTE (12): RETURN ON INVESTMENT (ROI)



a) Weighting Question	b) 'As-it-was' Question	c) 'As-it-is' Question
4	4	4

Figure 60: The answers for the quality attribute ROI (*Attachment 6: UBS answers*)

The approach of calculating an expected time range and financial benefits from a certain amount of invested resources, all depends on the formula of the return on investment (ROI). Having a formula that is adjusted to the architecture that it is being applied on is as it seems of great interest for UBS. With the answer 'The company is aware of the risks of not adjusting the ROI according to IT-architecture and is therefore very precise with ROI calculations.' on the question 'Does the company adjust ROI calculations along with changing IT-architecture(s)?', UBS states the essence.

However, having received a 'Yes, I strongly agree' for both the old and new architecture indicates that the question 'Are the resources invested in the new architecture calculated in terms of educed integration costs, improved asset reuse, and/or greater business agility?' has been misinterpreted. After all, a ROI for a silo architecture should not have been calculated in behalf of integration costs, asset reuse and business agility, as this cannot be provided by such a kind of architecture. On the other hand, UBS might at that point in time have considered the silo-based architecture as being able to provide integration, reusability and business agility. Assuming this, the answer is correctly given, but maybe not correctly formulated. In other words, the answer 'The architecture is a SOA and was implemented on behalf of an appropriate ROI.', should not mention the word SOA, but more precisely focus on the fact that this answer alternative should be chosen when the architecture was implemented on behalf of a suitable ROI.

One way to avoid the problem of knowing to what extent UBS has been using the most applicable ROI would have been to let UBS specify the formula being used. Furthermore, it would have been of interest to know what approach UBS used when discussing whether or not the company should invest in an architecture like the service-oriented one.

In summary, the ROI seems to be of great essence for the business when discussing potential architecture. Furthermore, the ROI-formula of UBS is said to be adjustable in accordance with the architecture being considered.

EVALUATION OF THE SOA QUALITY EVALUATION MODEL

As explained earlier in the chapters *Theoretical framework* and *Methodological approach* for example all the 12 quality attributes are divided into being either technical or business oriented attributes. Looking at the SOA Quality Evaluation Model, this is not visualized. Hence, this differentiation has only been applied to clarify the approach and foundation of the attribute gathering. Furthermore, as the model is a Spider web combining all its parts closely with each other, it was even regarded as inappropriate to make a distinction between technical and business as these in fact are to be combined in SOAs. Still wanting to separate the two perspectives, one could of course apply some kind of shading in the model.

Focusing on the attributes being chosen, it has already been discussed whether or not they are suitable quality attributes and most of all SOA quality attributes. Extending this discussion a bit further, the questions about to what extent the used attributes were in accordance with what one would be expecting of such a model and whether or not the model considered all necessary attributes arises. Due to the circumstances and the chosen approach of the research, these questions are unfortunately not possible to answer. The model per se did simply not go through an evaluation, but rather the results being gained. Having this in mind one improvement for future research would be to:

- focus less on the actual model results and pay greater attention to the model, i.e. guarantee for instance that all possible attributes are present

Another aspect that has not been mentioned is the interaction between attributes. Considering that all attributes are chosen separately from each other and only in terms of SOA and the two perspectives, there cannot be any specific interaction. In fact the attributes were deliberately chosen to be as little as possible related to each other. The overall outcome was simply thought to involve the extent of a service-oriented architecture being positive or negative for UBS. This in turn, was considered to be fulfilled by looking at the interception of the ‘As-it-is’ and ‘As-it-was’ lines at each separate attribute. After all, considering each attribute separately also contributes to an overall picture. Still, a possible improvement would be to:

- place the attributes accordingly to each other, i.e. take advantage of possible relationships, to get a descriptive overview

Speaking about connectivity and dimensions or perspectives one could also consider including the weighting in the in the ‘As-it-as’ and As-it-is” lines. This would take away the, at the moment, unmated weighting circles and integrate the aspect more into the overall model.

The fact that four answers received were marked with ‘I neither agree nor disagree’, caused some confusion in trying to explain the reason for the chosen answer, see the discussions for attribute modifiability, portability, integrability and especially efficiency. To avoid this confusion, the following improvements should be considered:

- the formulation of the questions and answers. ‘I neither agree nor disagree’, as an answer is not providing any specific feedback. On the other hand, the answer might not be the problem, but rather that the questions are incorrectly interpreted.

- Adding some qualitative questions and thus making it possible to receive information about why certain quantitative answers were provided. The answer for the 'Usability' attribute in the 'As-it-was' state e.g. did not match the expectations.

A final aspect to be discussed is to what extent the model is applicable to SOA implementations. This question leads once again back to the discussions being held in the chapters *Technical perspective* and *Business perspective*, i.e. due to the given assumptions eight of the twelve chosen attributes are considered to be obvious SOA attributes and three potential ones. Hence, as only one attribute is declared as less appropriate, the model per se clearly represents SOA features. Due to theses circumstances it should also be appropriate to assume that the model is applicable to SOA implementations. With the aim of being able to generalize this conclusion on behalf of research results, the model would have to be:

- applied on either other business areas or banking areas, in e.g. other countries. Furthermore, it would be of interest to add a question addressing whether or not the respondent considers the model as being applicable to SOA implementations.

CONCLUSION

UP UNTIL NOW THE READER HAS BEEN PROVIDED WITH A THEORETICAL SECTION, CONTAINING THE ESSENTIAL ASPECTS OF SOA, UBS AND QUALITY ATTRIBUTES, THE APPROACH BEING CONDUCTED TO GAIN THE NEEDED FEEDBACK AT UBS, AS WELL AS THE RESULTS BEING GATHERED. IN A FINAL ATTEMPT TO PICTURE THE OUTCOME OF THIS RESEARCH, THIS SUMMARIZING AND CONCLUDING CHAPTER WAS ESTABLISHED.

Many of the architectural discussions and proposals, currently being held and formulated at IBM, focus on SOA. In fact, every other day, IBM employees receive mails containing SOA information. Having in mind that IBM is one of the leading consultant and software companies in the world, it simply has to be realized that the architecture being implemented at UBS is a vital part in the technical evolution.

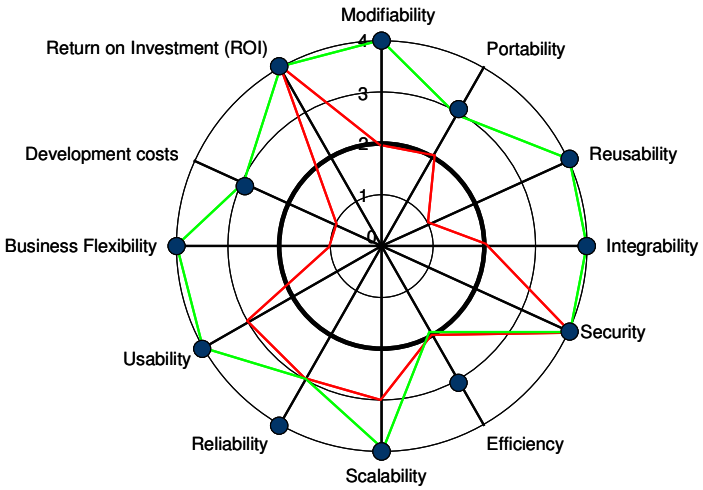
As the aim of this thesis has been to create a SOA Quality Evaluation Model that is applicable to SOA implementations, as well as applying this model on UBS to evaluate the impact of the newly implemented service-based architecture, the conclusion of this thesis will evolve around these two aspects.

THE SOA QUALITY EVALUATION MODEL

As can be deduced from the results being gathered and the analysis being conducted, the SOA Quality Evaluation Model was successfully created and applied at UBS. Moreover, the twelve selected attributes, based on mainly secondary data, were proven to be a combination of obvious and potential, with one minor deviance that turned out to be less appropriate, SOA quality attributes. Due to this, the answers being received for the old architecture and despite some discrepancies it can be concluded that the model is applicable to SOA implementations, as well as old non-service-oriented architectures.

Some of the discrepancies that occurred are unfortunately to be traced back to starting point of this research, i.e. to the date where the first discussions about the aim and interaction between university and business took place. Having fulfilled the research, the aspect of not having been able to fully provide answers on the research questions, is somewhat unsatisfactory. Hence, the focus on presenting the differences between the old and the new architecture at UBS gained more in interest than the actual creation of the model. This is definitely an aspect, showing the difficulty in conducting a research in combination with a university and a business. The interest of both parties simply differs greatly and has to be very carefully managed.

UBS AND ITS NEW ARCHITECTURE



As being mentioned in the section *Evaluation of the SOA Quality Evaluation Model*, the overall picture of the model clearly presents where a SOA has reached a beneficial or limited level of quality for the business, i.e. UBS. What is does not is to describe what interaction there is between the attributes and to what extent this interaction generates a specific overall output. Hence, the interesting aspect of this conclusion is based on how each separate quality attribute contributes or limits the success of UBS.

In summary, all attributes, except 'Security', 'Efficiency', 'Reliability' and 'ROI', indicated an improvement with the replacement of Abacus and none of the 12 quality measurements showed proof of deterioration. Moreover, nine out of the 12 attributes were regarded as critical for success. Hence, in general it can be assumed that these attributes have been in accordance with what can be expected from an architecture evaluation model at a bank like UBS.

Some of the interesting results being gained evolve around the fact that the new architecture in eight out of nine times achieved the maximum level of quality, being stated as vital for business agility by UBS. The greatest differences between the old and new architecture turned out to be within 'Reusability', 'Business flexibility' and to some extent 'Development cots', which all underline typical features for component-oriented architectures and therefore should show great differences when the comparison is conducted between a silo-based and a service-oriented architecture.

Finally, as this research has been shown, this architecture replacement should be nothing but of great use for UBS, both on the technical and business oriented side. Surely, the Front-End is still not yet implemented and also some other minor implementations are still to be conducted, but these details should not be affecting the overall positive result.

FUTURE RESEARCH SUGGESTIONS

As a sum-up of the discussion being held in section *Evaluation of the SOA Quality Evaluation Model* the following aspects would be of great use to consider when wanting to extend or improve this model.

- Extend the scientific foundation of SOA quality attributes, by conducting a research where several companies are asked to state what they would include in a SOA Quality Evaluation Model.

- Create connectivity of the quality attributes, i.e. state how the different attributes can be put in relationship to each other and integrate the weighting aspect in the lines showing the differences between the old and new architecture.

- Re-formulate questions and answers, where a 'I neither agree nor disagree' and/or confusing answers were provided, or apply a more open research method, where the researcher is allowed to support the respondent or where qualitative questions lead to an avoidance of possible misunderstanding.

- Apply the model on other research areas than banks. Doing this might also lead to a need of changing the questions, even though they are intended to be applicable to other research areas as well.

ATTACHMENTS

ATTACHMENT 1: OVERVIEW MATRIX OF POTENTIAL QUALITY ATTRIBUTES

Potential Quality Attributes	ISO/IEC 9126 (1994)	Bass et. al. (1998)	CUPRIMDSO *(Bencher, 1994)*	FURPS *(Grady & Caswell, 1987)*	Kan (2003)	McCall (1977)	Boehm (1978)	Sum	UBS articles	Business articles	SOA articles	Sum	Total
Technical perspective													
Modifiability		1				1		2			1	1	3
Maintainability	1	1	1	1			1	5			1	1	6
Stability	1							1				0	1
Analysability	1							1				0	1
Changeability	1	1						2				0	2
Testability	1			1				2			1	1	3
Supportability				1				1				0	1
Adaptability				1				1				0	1
Compatibility				1				1				0	1
Configurability								0				0	0
Localizability								0				0	0
Capability			1					1				0	1
Functionality	1	1		1				3				0	3
Security	1	1						2	1		1	2	4
Integrity						1		1				0	1
Suitability	1							1				0	1
Accuracy	1							1				0	1
Interoperability	1					1		2				0	2
Integrability		1						1	1		1	2	3
Installability			1	1				2				0	2
Efficiency	1					1	1	3			1	1	4
Performance		1	1	1				3				0	3
Reliability	1		1	1		1	1	5			1	1	6
Maturity	1							1				0	1
Fault tolerance	1							1				0	1
Recoverability	1							1			1	1	2
Availability		1						1			1	1	2
Portability	1	1				1	1	4				0	4
Adaptability	1							1			1	1	2
Replaceability	1							1			1	1	2
Reusability		1				1		2	1	1	1	3	5
Scalability		1		1				2	1		1	2	4
Business perspective													
Documentation			1					1				0	1
Business efficiency								0	1			1	1
Serviceability			1					1				0	1
Business qualities								0				0	0
Time to market		1						1	1	1	1	3	4
System lifetime		1						1				0	1
Targeted market		1						1				0	1
Revenue								0	1		1	2	2
Return on Investment (ROI)								0	1	1	1	3	3
Integration expense								0	1	1		2	2
Asset reuse								0		1	1	2	2
Business agility								0		1		1	1
Business risk								0		1		1	1
Flexibility								0	1	1	1	3	3
Development time								0		1	1	2	2
Development cost								0	1	1	1	3	3
Obsolescence								0			1	1	1
Competition								0		1		1	1
Usability	1	1	1	1	1			5		1	1	2	7
Understandability	1							1				0	1
Learnability	1							1				0	1
Operateability	1							1				0	1
Customer satisfaction			1		1			2	1			1	3
Project time					1			1			1	1	2
Project costs								0			1	1	1

ATTACHMENT 2: BUNDLED OVERVIEW MATRIX OF POTENTIAL QUALITY ATTRIBUTES

Potential Quality Attributes	**ISO/IEC 9126** (1994)	**Bass et. al.** (1998)	**CUPRIMDSO** *(Bencher, 1994)*	**FURPS** *(Grady & Caswell, 1987)*	**Kan** (2003)	**McCall** (1977)	**Boehm** (1978)	Sum	UBS articles	Business articles	SOA articles	Sum	Total
Technical perspective													
Modifiability	2 (Maintainability; Changeability)	3 (Maintainability; Changeability)	1 (Maintainability)	2 (Maintainability)		1	1 (Maintainability)	10			1 (Maintainability)	1	11
Stability	1							1				0	1
Analysability	1							1				0	1
Testability	1			1				2			1	1	3
Supportability				1				1				0	1
Adaptability				1				1				0	1
Compatibility				1				1				0	1
Configurability								0				0	0
Localizability								0				0	0
Functionality	1	1	1 (Capability)	1				4				0	4
Security	1	1				1 (Integrity)		2	1		1	2	4
Suitability	1							1				0	1
Accuracy	1							1				0	1
Integrability	1 (Interoperability)	1	1 (Installability)	1 (Installability)		1 (Interoperability)		5	1		1	2	7
Efficiency	1	1 (Performance)	1 (Performance)	1 (Performance)		1	2 (Performance)	7	1 (Business efficiency)		1	2	9
Reliability	2 (Recoverability; Availability)	1 (Availability)	1	1		1	1	7			3 (Availability; Recoverability)	3	10
Maturity	1							1				0	1
Fault tolerance	1							1				0	1
Portability	2 (Replaceability)	1				1	1	5			1 (Replaceability)	1	6
Reusability		1				1		2	1	2 (Asset reuse)	2 (Asset reuse)	5	7
Scalability		1		1				2	1		1	2	4
Business perspective													
Documentation			1					1				0	1
Serviceability			1					1				0	1
Business qualities								0				0	0
System lifetime		1						1				0	1
Targeted market		1						1				0	1
Return on Investment (ROI)		1 (Time to market)						1	3 (Time to market; Revenue)	2 (Time to market)	3 (Time to market; Revenue)	8	9
Business risk								0		1		1	1
Business flexibility	1 (Adaptability)							1	1	2 (Business agility; Flexibility)	2 (Adaptability)	5	6
Development cost					1 (Project time)			1	2 (Integration expenses)	2 (Development time)	4 (Project costs; Project time; Development time)	8	9
Obsolescence								0			1	1	1
Usability	3 (Learnability; Understandability)	1	1	1	1			7		1	1	2	9
Operateability	1							1				0	1
Customer satisfaction			1		1			2	1			1	3

ATTACHMENT 3: OVERVIEW MATRIX OF SELECTED QUALITY ATTRIBUTES

Quality Attributes	UBS articles	Business articles	SOA articles	Sum	Total
Techical perspective					
Modifiability			1 (Maintainability)	1	11
Stability				0	1
Analysability				0	1
Testability			1	1	3
Supportability				0	1
Adaptability				0	1
Compatibility				0	1
Configurability				0	0
Localizability				0	0
Functionality				0	4
Security	1		1	2	4
Suitability				0	1
Accuracy				0	1
Integrability	1		1	2	7
Efficiency	1 (Business efficiency)		1	2	9
Reliability			3 (Availability; Recoverability)	3	10
Maturity				0	1
Fault tolerance				0	1
Portability			1 (Replaceability)	1	6
Reusability	1	2 (Asset reuse)	2 (Asset reuse)	5	7
Scalability	1		1	2	4
Business perspective					
Documentation				0	1
Serviceability				0	1
Business qualities				0	0
System lifetime				0	1
Targeted market				0	1
Return on Investment (ROI)	3 (Time to market; Revenue)	2 (Time to market)	3 (Time to market; Revenue)	8	9
Business risk		1		1	1
Business flexibility	1	2 (Business agility; Flexibility)	2 (Adaptability)	5	6
Development cost	2 (Integration expenses)	2 (Development time)	4 (Project costs; Project time; Development time)	8	9
Obsolescence			1	1	1
Usability		1	1	2	9
Operateability				0	1
Customer satisfaction	1			1	3

ATTACHMENT 4: INTRODUCTION FOR THE RESPONDENT

Dear respondent.

By filling out this questionnaire, you are helping me with my Diploma thesis, written at the Swedish University of Lund – (Lunds Universitet, Schools of Economics, Department: Informatics) – in cooperation with IBM Switzerland.

Since the aim of the thesis is to create a service-oriented architecture (SOA) quality evaluation model that is applicable for SOA implementations, I need input from companies that already have managed to gather experience within this area. For these reasons, you and your company have been selected.

As I am now approaching the final part of my thesis I would be most grateful, if you could fill out the questionnaire by **14th of July 2006**, the latest, and send it back to me to one of the following addresses: **xxx@hermes.ics.lu.se** or **xxx@ch.ibm.com**

If you should have any questions while you are filling out the questionnaire, please do not hesitate to contact me on the email addresses mentioned above or on my mobile phone: **xxx xxx xx xx**

Instructions

1. Before you open the Excel-sheet, please answer the following:

Company name: _____

Working position: _____

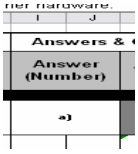
Years of employment at the company: _____

Knowledge about: ☐ The old architecture ☐ The new architecture

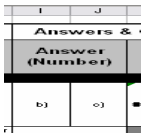
Do you want your answers to be treated confidentially: ☐ Yes ☐ No

2. Having opened the Excel-sheet, please do the following:

- Answer all questions marked with "a)" by putting a value from 0 to 4 in the columns "J" and "K", where the "a)" is placed.



- Answer all questions marked with "b+c)" by putting a value from 0 to 4 in the columns "J" and "K", where "b)" and "c)" are placed respectively. "b)" is to present the "as-it-was"-state, before the new architecture was implemented. "c)" on the other hand, asks the respondent for the experience with the new architecture.



- To be able to find out what specific value the answer that is the most applicable one has, row three presents the alternatives in brackets.

2	SOA Quality Attributes		Questions	Likert scale,
3	Attribute	Description	a) Weightening Question b+c) "As-it-was" and "As-it-is" Question	Yes, I agree (3)
4	Technical perspective			
	1] Modifiability	Modifiability touches the ability of changing a set of	1a] Is one of the companies target goals to be able to	Modifiability is important, but not vital.

Thank you for your participation!

Kind regards, Annika Pettersson

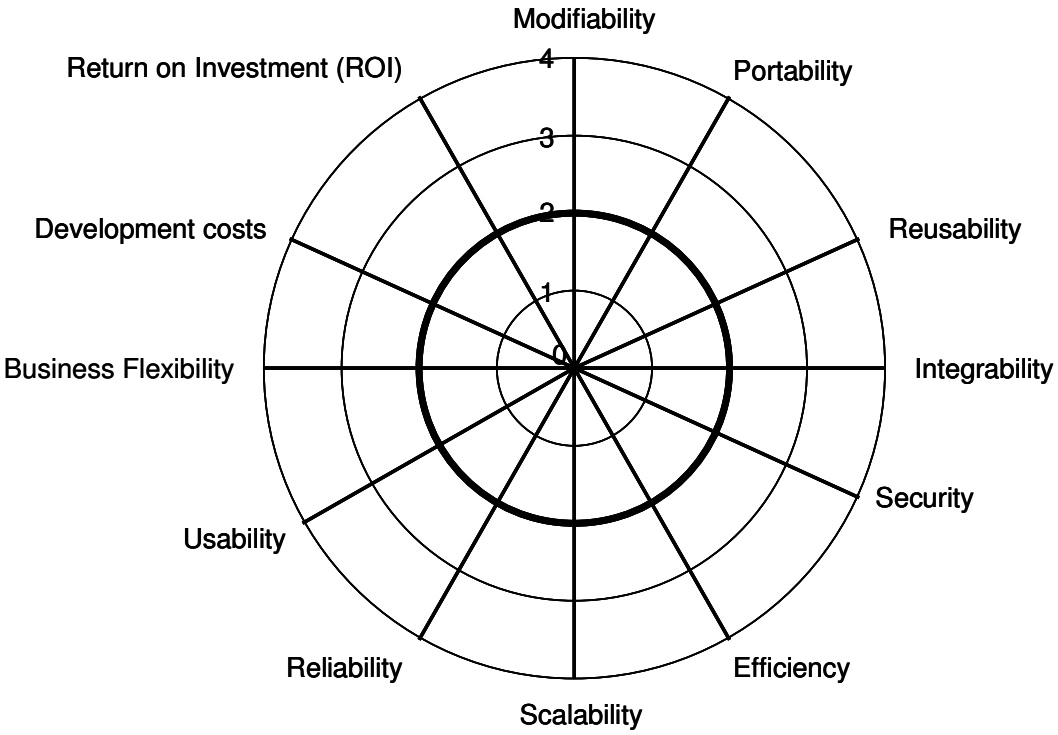
ATTACHMENT 5: QUESTIONNAIRE - SOA QUALITY ATTRIBUTES

SOA Quality Attributes		Questions	Likert scale, Answer alternatives					Answers & Calculations			
Attribute	**Description**	**a) Weighting Question b+c) "As-it-was" and "As-it-is" Question**	**Yes, I strongly agree (4)**	**Yes, I agree (3)**	**I neither agree nor disagree (2)**	**No, I disagree (1)**	**No, I strongly disagree (0)**	**Answer (Number)**		**Weigthening**	
Technical perspective											
1) Modifiability	Modifiability touches the ability of changing a set of attributes (e.g. processes, technologies, components, services, rules etc.) within an architecture without actually influencing the overall architectural structure. Modifications considered are: · Deleting unwanted capabilities. · Restructuring · Adapting to new operating environments. · Extending or adding attributes	**1a)** Is one of the companies target goals to be able to modify attributes within the company, without increasing the complexity and rearranging the structure of the architecture?	Modifiability is vital for the business. (Modifiability is a critical success factor (CSF).	Modifiability is important, but not vital.	The company does not make any statement about this attribute.	The company does not consider modifiability as being important, i.e. modifications are to lead to new architectural structures.	Modifiability is of no interest whatsoever for the company.	**a)**			
		1b+c) Is the architecture open for attribute modifications, as business situations change?	The architecture supports business agility and has an open structure.	The architecture is relatively flexible and allows most modifications.	The question cannot be answered.	The architecture does not support changes, without affecting the architectural structure.	The architecture has a closed and/or extremely complex architectural structure and thus cannot support any modifications.	b)	c)	#WERT!	#WERT!
2) Portability	The opportunity of adapting to different environments (e.g. platforms, countries) and considering cultural internationalization (e.g. language and legislative adjustments and/or specific system preferences) without having to make major architectural changes.	**2a)** Does the company see portabilities such as being able to update the IT-architecture with the latest hardware and/or to transfer the architecture to other areas, as vital?	Portability is vital for the business (Portability is a CSF.)	Portability is important, but not vital.	The company does not make any statement about this attribute.	To have a portable architecture is of no greater interest for the company.	Portability is of no interest whatsoever for the company.	**a)**			
		2b+c) Is the architecture suitable for adapting to new environments?	The architecture is highly portable in terms of platform independency and cultural internationalization.	The architecture is relatively portable and supports limited platform independency. Cultural internationalization, however, is not supported.	The question cannot be answered.	As platforms changes or movements to other countries take place, the architecture has to go through severe changes.	The architecture does not provide any portability whatsoever, i.e. not even with some architectural changes the system will be usable in terms of portability.	b)	c)	#WERT!	#WERT!
3) Reusability	The ability to reuse some of the system's structure or components in other systems or the same system.	**3a)** Is one of the target goals to implement an architecture, where a specific function only is implemented once and reused all other times?	To reuse already available resources is a vital process within the company, i.e. components should constantly be reused when possible. (Reusability is a CSF.)	Reusability is important, but not vital. Components should be reused, but directions of this kind are not formally given by the stab.	The company does not make any statement about this attribute.	Reusing already existing architectural components is of no greater interest for the company, i.e. in some cases similar components occur in the architecture.	The company uses a silo-based perspective and is thus not interested in reusing components.	**a)**			
		3b+c) Does the architecture support the reusing of already existing structures and components?	In the architecture a function that is the same in one or more business areas is only implemented once in the whole company.	In the architecture components are reused, but are slightly adjusted to the new circumstances.	The question cannot be answered.	Similar components and structures tend to occur in the architecture, due to lack of reusability.	In the architecture all functions are implemented separately for each business, unrelated to other business areas and already existing material.	b)	c)	#WERT!	#WERT!
4) Integrability	To integrate single components with other ones or groups of components with other groups or systems.	**4a)** Is it vital for the ongoing business that parts of the architecture can interact with other parts via well defined interfaces?	Without a properly working integration between architectural components, the business will not even be able to perform the easiest business processes. (Integrability is a CSF.)	Having a properly functioning architecture is important, but not all components have to be integrated optimally to continue with the high level of business activities.	The company does not make any statement about this attribute.	If some problems regarding the integration occur now and then, this does not bother the company.	Constantly occurring complications, due to integration, are of no interest whatsoever for the company.	**a)**			
		4b+c) Does the architecture provide interaction between architectural parts through an interface language, a middleware (e.g. ESB), a directory, protocols and/or interfaces?	The new architecture integrates its components/services through interfaces over a middleware with protocols.	The new architecture integrates some components/services through interfaces over a middleware with protocols, others not.	The question cannot be answered.	The architecture does not support integrability between components and services.	The architecture does not need integration capabilities.	b)	c)	#WERT!	#WERT!
5) Security	The ability to resist unauthorized attempts of usage and denial of service while still providing services to legitimate users.	**5a)** Are security actions critical for the overall business operations?	Functional security is vital for the business. (Security is a CSF.)	Security is important, but not vital.	The company does not make any statement about this attribute.	Security is of minor importance, since the company manages fine without putting effort into this aspect.	Security aspects are of no importance whatsoever for the company, i.e. the architecture does not consider security.	**a)**			
		5b+c) Does the architecture include authentication checks, authorization, access controls and/or provide firewalls between networks and use digital signatures for the exchange of messages?	The architecture provides the involved parties with very good security, including network protection.	The architecture provides security, but not on a web basis, i.e. authentication checks, authorization and access controls take place, but not network protections.	The question cannot be answered.	The architecture does not provide enough security, i.e. basic aspects such as authentication checks, authorization or access controls are not present.	The architecture provides no security at all.	b)	c)	#WERT!	#WERT!
6) Efficiency	The relationship between the level of performance of the system and the amount of resources used.	**6a)** Is IT efficiency important for the company's business agility?	Efficiency is vital for achieving business agility. (Efficiency is a CSF.)	Efficiency is important, but not vital for the business.	The company does not make any statement about this attribute.	Efficiency does not affect the company's business agility.	Efficiency is completely unimportant for the company.	**a)**			
		6b+c) Does the architecture act efficiently with the available amount of hardware, business processes and generated performance?	The architecture has, amongst others, an optimal level of throughput, response time and recovery time and thus shows proof of great efficiency.	The architecture has, amongst others, a usable level of throughput, response time and recovery time and thus shows proof of good efficiency.	The question cannot be answered.	The architecture does not provide the company with enough efficiency.	The architecture is completely inefficient.	b)	c)	#WERT!	#WERT!
7) Scalability	The extent to which an architecture or system is able improve its performance proportionally to the adding of more load, i.e. more users, more data, more transactions and/or other hardware.	**7a)** Is scalability a critical success factor (CSF) for the company?	Scalability is critical for the success of the company. (Scalability is a CSF.)	Scalability is important, but not vital for the success of the company.	The company does not make any statement about this attribute.	Additional change of load in the system is not expected to lead to better performance, i.e. scalability is	Scalability is of no interest whatsoever for the company.	**a)**			
		7b+ c) Does the architecture support the adding of more loads by increasing the performance proportionally to the input?	The architectures provides a proportional increase in performance, with adding of more load. Thus the architecture maintains excellent scalability features.	The architectures provides an increase, but not proportional, in performance, when adding more load. Thus the architecture maintains scalability features.	The question cannot be answered.	The performance will decrease when the architecture is extended. Thus scalability is not present.	The architecture cannot be extended and thus performance will be unaffected no matter what and scalability cannot be measured.	b)	c)	#WERT!	#WERT!
8) Reliability	The ability of a system to keep a certain performance, with high availability and recoverability of architectural components, under stated conditions.	**8a)** Are the systems' reliability essential for the survival of the company?	Reliability is vital for the survival of the business. (Reliability is a CSF.)	Reliability is important, but not vital, for the business.	The company does not make any statement about this attribute.	Risk analyses are not conducted at the company.	Reliability is of no importance for the business.	**a)**			
		8b+c) Does the architecture support systems with high reliability?	The systems in the architecture all keep a high level of reliability, due to high availability of architectural components and good interaction between these components.	The architecture is relatively reliable, since safe messaging/packaging (interaction between components) is assured.	The question cannot be answered.	The architecture is unreliable, since messages/packages are not properly transferred. Hence, interaction is restricted.	The architecture completely unreliable, since availability, disaster recovery, back-up abilities and interaction between components is extremely low.	b)	c)	#WERT!	#WERT!
Business perspective											
9) Usability	The way users experience the systems interface, i.e. how easy it is to use, learn, memorize, understand and appreciate.	**9a)** Is interaction between users and systems essential for the overall culture of the company?	Satisfied and challenged system users is vital for the company culture. (Usability is a CSF.)	Satisfied and challenged system users is important, but not vital.	The company does not make any statement about this attribute.	System usability does not affect the overall business culture.	The company takes no interest whatsoever in creating a symbiosis between the system(s) and the user(s).	**a)**			
		9b+c) Are users of the systems generally speaking satisfied with the ease of use of the user interface?	The architecture supports the development of systems that are easy to learn, understand and memorize. Furthermore the system serves the users needs.	The architecture supports systems that serve the users needs or are easy to learn, understand or memorize.	The question cannot be answered.	The system is not easy to learn, understand and memorize, and it does not serve the users needs.	The architecture does not consider human computer interaction (HCI) at all.	b)	c)	#WERT!	#WERT!
10) Business Flexibility	To be able to adjust quickly to new market demands, due to optimized business processes and products.	**10a)** Is business flexibility one of the CSFs of the company?	Flexibility of business processes and products is critical for the success of the company. (Flexibility is a CSF.)	Flexibility of business processes and products is important for the success of the company.	The company does not make any statement about this attribute.	Flexibility of business processes and products is not important for the success of the company.	Flexibility of business processes and products does not interest the company.	**a)**			
		10b+c) Are the business processes and products in the architecture flexible enough to adjust quickly to new market demands?	The architecture provides extraordinary opportunities for gaining competitive advantages.	The new architecture provides relatively good opportunities for gaining competitive advantages.	The question cannot be answered.	The architecture provides poor opportunities for gaining competitive advantages.	The architecture is not appropriate for competition.	b)	c)	#WERT!	#WERT!
11) Development costs	The costs created through the development phase of a system.	**11a)** Is keeping low costs, when extending the overall business architecture, a part of the business' IT-strategy?	Having an architecture with reusable components and optimized business processes is a part of the overall IT-strategy. (Low Development costs are CSFs.)	Having an architecture with reusable components and optimized business processes is important for the business.	The company does not make any statement about this attribute.	Reduced development costs are not a part of the IT-strategy.	Reduced development costs are of no importance.	**a)**			
		11b+c) Can costs be kept low due to optimized business processes and reusable IT components, when new systems are developed within architecture?	The architecture provides the business with the possibility to implement new systems to low costs.	The architecture provides the business with the possibility to implement new systems to relatively low costs.	The question cannot be answered.	The architecture does not provide cost efficiency, i.e. low costs might have been kept, but not due to reusability and efficient processes.	The architecture cannot be used for the implementation of new systems, due to completely unrealistic high development costs.	b)	c)	#WERT!	#WERT!
12) Return on Investment (ROI)	The approach of calculating an expected time range and financial benefits from investing a certain amount of resources.	**12a)** Does the company adjust ROI calculations along with changing IT-architecture(s)?	The company is aware of the risks of not adjusting the ROI according to IT-architecture and is therefore very precise with ROI calculations.	The company adjust the ROIs according to the architectures being considered, but does not go further into what is specific for each architecture. Thus, the ROI will only be relatively applicable.	The company does not make any statement about this attribute.	Having the correct ROI calculation is of no greater importance for the company.	The company does not use ROI calculations as a supportive tool.	**a)**			
		12b+c) Are the resources invested in the architecture calculated in terms of educed integration costs, improved asset reuse, and/or greater business agility?	The architecture is a SOA and was implemented on behalf of an appropriate ROI.	The architecture has some SOA features and thus the ROI is only partially adjusted.	The question cannot be answered.	The ROI calculation maintained other factors, than suitable for the concerned architecture.	The new architecture is no SOA, according to the classical ROI used.	b)	c)	#WERT!	#WERT!

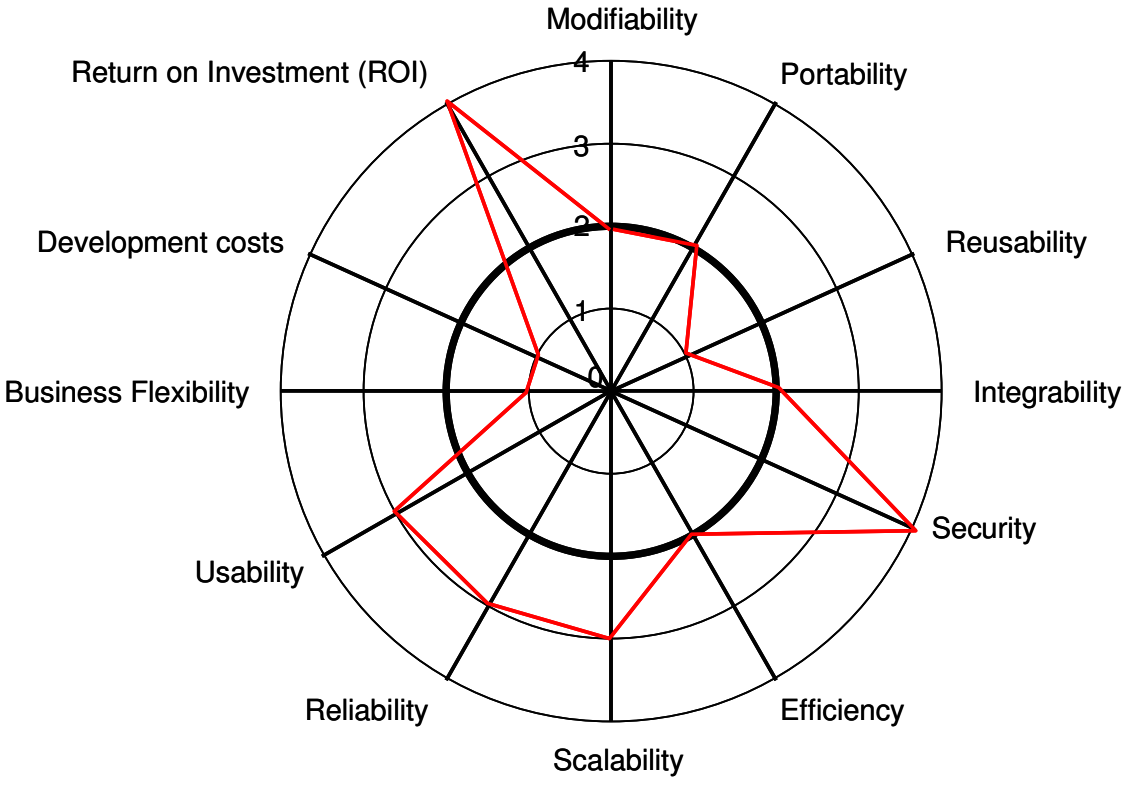
ATTACHMENT 6: UBS ANSWERS

SOA Quality Attributes		Questions	Likert scale, Answer alternatives					Answers & Calculations	
Attribute	**Description**	**a) Weightening Question b+c) "As-it-was" and "As-it-is" Question**	**Yes, I strongly agree (4)**	**Yes, I agree (3)**	**I neither agree nor disagree (2)**	**No, I disagree (1)**	**No, I strongly disagre (0)**	**Answer (Number)**	
Technical perspective									
1) Modifiability	Modifiability touches the ability of changing a set of attributes (e.g. processes, technologies, components, services, rules etc.) within an architecture without actually influencing the overall architectural structure. Modifications considered are: · Deleting unwanted capabilities. · Restructuring · Adapting to new operating environments. · Extending or adding attributes	**1a)** Is one of the companies target goals to be able to modify attributes within the company, without increasing the complexity and rearranging the structure of the architecture?	Modifiability is vital for the business. (Modifiability is a critical success factor (CSF).)	Modifiability is important, but not vital.	The company does not make any statement about this attribute.	The company does not consider modifiability as being important, i.e. modifications are to lead to new architectural structures.	Modifiability is of no interest whatsoever for the company.	**4**	
		1b+c) Is the architecture open for attribute modifications, as business situations change?	The architecture supports business agility and has a open structure.	The architecture is relatively flexible and allows most modifications.	The question cannot be answered.	The architecture does not support changes, without affecting the architectural structure.	The architecture has a closed and/or extremely complex architectural structure and thus cannot support any modifications.	2	4
2) Portability	The opportunity of adapting to different environments (e.g. platforms, countries) and considering cultural internationalization (e.g. language and legislative adjustments and/or specific system preferences) without having to make major architectural changes.	**2a)** Does the company see portabilities such as being able to update the IT-architecture with the latest hardware and/or to transfer the architecture to other areas, as vital?	Portability is vital for the business (Portability is a CSF.)	Portability is important, but not vital.	The company does not make any statement about this attribute.	To have a portable architecture is of no greater interest for the company.	Portability is of no interest whatsoever for the company.	**3**	
		2b+c) Is the architecture suitable for adapting to new environments?	The architecture is highly portable in terms of platform independency and cultural internationalization.	The architecture is relatively portable and supports limited platform independency. Cultural internationalization, however, is not supported.	The question cannot be answered.	As platforms changes or movements to other countries take place, the architecture has to go through severe changes.	The architecture does not provide any portability whatsoever, i.e. not even with some architectural changes the system will be usable in terms of portability.	2	3
3) Reusability	The ability to reuse some of the system's structure or components in other systems or the same system.	**3a)** Is one of the target goals to implement an architecture, where a specific function only is implemented once and reused all other times?	To reuse already available resources is a vital process within the company, i.e. components should constantly be reused when possible. (Reusability is a CSF.)	Reusability is important, but not vital. Components should be reused, but directions of this kind are not formally given by the stab.	The company does not make any statement about this attribute.	Reusing already existing architectural components is of no greater interest for the company, i.e. in some cases similar components occur in the architecture.	The company uses a silo-based perspective and is thus not interested in reusing components.	**4**	
		3b+c) Does the architecture support the reusing of already existing structures and components?	In the architecture a function that is the same in one or more business areas is only implemented once in the whole company.	In the architecture components are reused, but are slightly adjusted to the new circumstances.	The question cannot be answered.	Similar components and structures tend to occur in the architecture, due to lack of reusability.	In the architecture all functions are implemented separately for each business, unrelated to other business areas and already existing material.	1	4
4) Integrability	To integrate single components with other ones or groups of components with other groups or systems.	**4a)** Is it vital for the ongoing business that parts of the architecture can interact with other parts via well defined interfaces?	Without a properly working integration between architectural components, the business will not even be able to perform the easiest business processes. (Integrability is a CSF.)	Having a properly functioning architecture is important, but not all components have to be integrated optimally to continue with the high level of business activities.	The company does not make any statement about this attribute.	If some problems regarding the integration occur now and then, this does not bother the company.	Constantly occurring complications, due to integration, are of no interest whatsoever for the company.	**4**	
		4b+c) Does the architecture provide interaction between architectural parts through an interface language, a middleware (e.g. ESB), a directory, protocols and/or interfaces?	The new architecture integrates its components/services through interfaces over a middleware with protocols.	The new architecture integrates some components/services through interfaces over a middleware with protocols, others not.	The question cannot be answered.	The architecture does not support integrability between components and services.	The architecture does not need integration capabilities.	2	4
5) Security	The ability to resist unauthorized attempts of usage and denial of service while still providing services to legitimate users.	**5a)** Are security actions critical for the overall business operations?	Functional security is vital for the business. (Security is a CSF.)	Security is important, but not vital.	The company does not make any statement about this attribute.	Security is of minor importance, since the company manages fine without putting effort into this aspect.	Security aspects are of no importance whatsoever for the company, i.e. the architecture does not consider security.	**4**	
		5b+c) Does the architecture include authentication checks, authorization, access controls and/or provide firewalls between networks and use digital signatures for the exchange of messages?	The architecture provides the involved parties with very good security, including network protection.	The architecture provides security, but not on a web basis, i.e. authentication checks, authorization and access controls take place, but not network protections.	The question cannot be answered.	The architecture does not provide enough security, i.e. basic aspects such as authentication checks, authorization or access controls are not present.	The architecture provides no security at all.	4	4
6) Efficiency	The relationship between the level of performance of the system and the amount of resources used.	**6a)** Is IT efficiency important for the company's business agility?	Efficiency is vital for achieving business agility. (Efficiency is a CSF.)	Efficiency is important, but not vital for the business.	The company does not make any statement about this attribute.	Efficiency does not affect the company's business agility.	Efficiency is completely unimportant for the company.	**3**	
		6b+c) Does the architecture act efficiently with the available amount of hardware, business processes and generated performance?	The architecture has, amongst others, an optimal level of throughput, response time and recovery time and thus shows proof of great efficiency.	The architecture has, amongst others, a usable level of throughput, response time and recovery time and thus shows proof of good efficiency.	The question cannot be answered.	The architecture does not provide the company with enough efficiency.	The architecture is completely inefficient.	2	2
7) Scalability	The extent to which an architecture or system is able improve its performance proportionally to the adding of more load, i.e. more users, more data, more transactions and/or other hardware.	**7a)** Is scalability a critical success factor (CSF) for the company?	Scalability is critical for the success of the company. (Scalability is a CSF.)	Scalability is important, but not vital for the success of the company.	The company does not make any statement about this attribute.	Additional change of load in the system is not expected to lead to better performance, i.e. scalability is	Scalability is of no interest whatsoever for the company.	**4**	
		7b+ c) Does the architecture support the adding of more load by increasing the performance proportionally to the input?	The architectures provides an proportional increase in performance, when adding of more load. Thus the architecture maintains excellent scalability features.	The architectures provides an increase, but not proportional, in performance, when adding more load. Thus the architecture maintains scalability features.	The question cannot be answered.	The performance will decrease when the architecture is extended. Thus scalability is not present.	The architecture cannot be extended and thus performance will be unaffected no matter what and scalability cannot be measured.	3	4
8) Reliability	The ability of a system to keep a certain performance, with high availability and recoverability of architectural components, under stated conditions.	**8a)** Are the systems' reliability essential for the survival of the company?	Reliability is vital for the survival of the business. (Reliability is a CSF.)	Reliability is important, but not vital, for the business.	The company does not make any statement about this attribute.	Risk analyses are not conducted at the company.	Reliability is of no importance for the business.	**4**	
		8b+c) Does the architecture support systems with high reliability?	The systems in the architecture all keep a high level of reliability, due to high availability of architectural components and good interaction between these components.	The architecture is relatively reliable, since safe messaging/packaging (interaction between components) is assured.	The question cannot be answered.	The architecture is unreliable, since messages/packages are not properly transferred. Hence, interaction is restricted.	The architecture completely unreliable, since availability, disaster recovery, back-up abilities and interaction between components is extremely low.	3	3
Business perspective									
9) Usability	The way users experience the systems interface, i.e. how easy it is to use, learn, memorize, understand and appreciate.	**9a)** Is interaction between users and systems essential for the overall culture of the company?	Satisfied and challenged system users is vital for the company culture. (Usability is a CSF.)	Satisfied and challenged system users is important, but not vital.	The company does not make any statement about this attribute.	System usability does not affect the overall business culture.	The company takes no interest whatsoever in creating a symbiosis between the system(s) and the user(s).	**4**	
		9b+c) Are users of the systems generally speaking satisfied with the ease of use of the user interface?	The architecture supports the development of systems that are easy to learn, understand and memorize. Furthermore the system serves the users needs.	The architecture supports systems that serve the users needs or are easy to learn, understand or memorize.	The question cannot be answered.	The system is not easy to learn, understand and memorize, and it does not serve the users needs.	The architecture does not consider human computer interaction (HCI) at all.	3	4
10) Business Flexibility	To be able to adjust quickly to new market demands, due to optimized business processes, products and/or suitable IT.	**10a)** Is business flexibility one of the CSFs of the company?	Flexibility of business processes, products and IT is critical for the success of the company. (Flexibility is a CSF.)	Flexibility of business processes, products and IT is important for the success of the company.	The company does not make any statement about this attribute.	Flexibility of business processes, products ad IT is not important for the success of the company.	Flexibility of business processes, products and IT does not interest the company.	**4**	
		10b+c) Are the business processes, products and IT in the architecture flexible enough to adjust quickly to new market demands?	The architecture provides extraordinary opportunities for gaining competitiveness advantages.	The new architecture provides relatively good opportunities for gaining competitive advantages.	The question cannot be answered.	The architecture provides poor opportunities for gaining competitive advantages.	The architecture is not appropriate for competition.	1	4
11) Development costs	The costs created through the development phase of a system.	**11a)** Is keeping low costs, when extending the overall business architecture, a part of the business' IT-strategy?	Having an architecture with reusable components and optimized business processes is a part of the overall IT-strategy. (Low Development costs are CSFs.)	Having an architecture with reusable components and optimized business processes is important for the business.	The company does not make any statement about this attribute.	Reduced development costs are not a part of the IT-strategy.	Reduced development costs are of no importance.	**3**	
		11b+c) Can costs be kept low due to optimized business processes and reusable IT components, when new systems are developed within architecture?	The architecture provides the business with the possibility to implement new systems to low costs.	The architecture provides the business with the possibility to implement new systems to relatively low costs.	The question cannot be answered.	The architecture does not provide cost efficiency, i.e. low costs might have been kept, but not due to reusability and efficient processes.	The architecture cannot be used for the implementation of new systems, due to completely unrealistic high development costs.	1	3
12) Return on Investment (ROI)	The approach of calculating an expected time range and financial benefits from investing a certain amount of resources.	**12a)** Does the company adjust ROI calculations along with changing IT-architecture(s)?	The company is aware of the risks of not adjusting the ROI according to IT-architecture and is therefore very precise with ROI calculations.	The company adjust the ROIs according to the architectures being considered, but does not go further into what is specific for each architecture. Thus, the ROI will only be relatively applicable.	The company does not make any statement about this attribute.	Having the correct ROI calculation is of no greater importance for the company.	The company does not use ROI calculations as a supportive tool.	**4**	
		12b+c) Are the resources invested in the architecture calculated in terms of educed integration costs, improved asset reuse, and/or greater business agility?	The architecture is a SOA and was implemented on behalf of an appropriate ROI.	The architecture has some SOA features and thus the ROI is only partially adjusted.	The question cannot be answered.	The ROI calculation maintained other factors, than suitable for the concerned architecture.	The new architecture is no SOA, according to the classical ROI used.	4	4

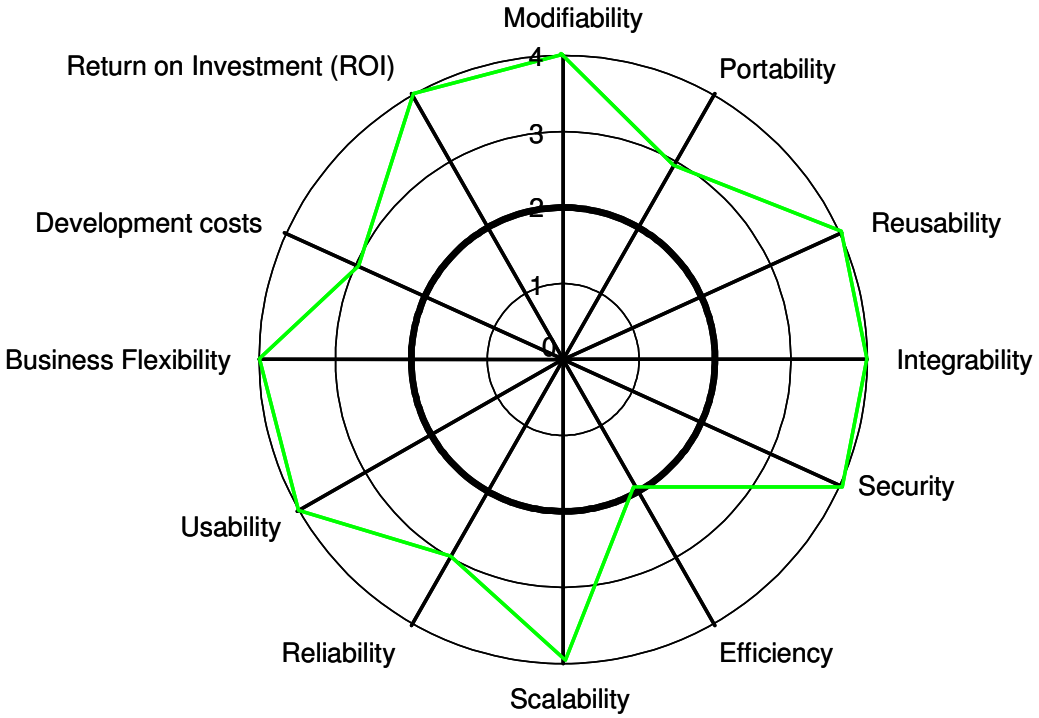
ATTACHMENT 7: THE SOA QUALITY EVALUATION MODEL



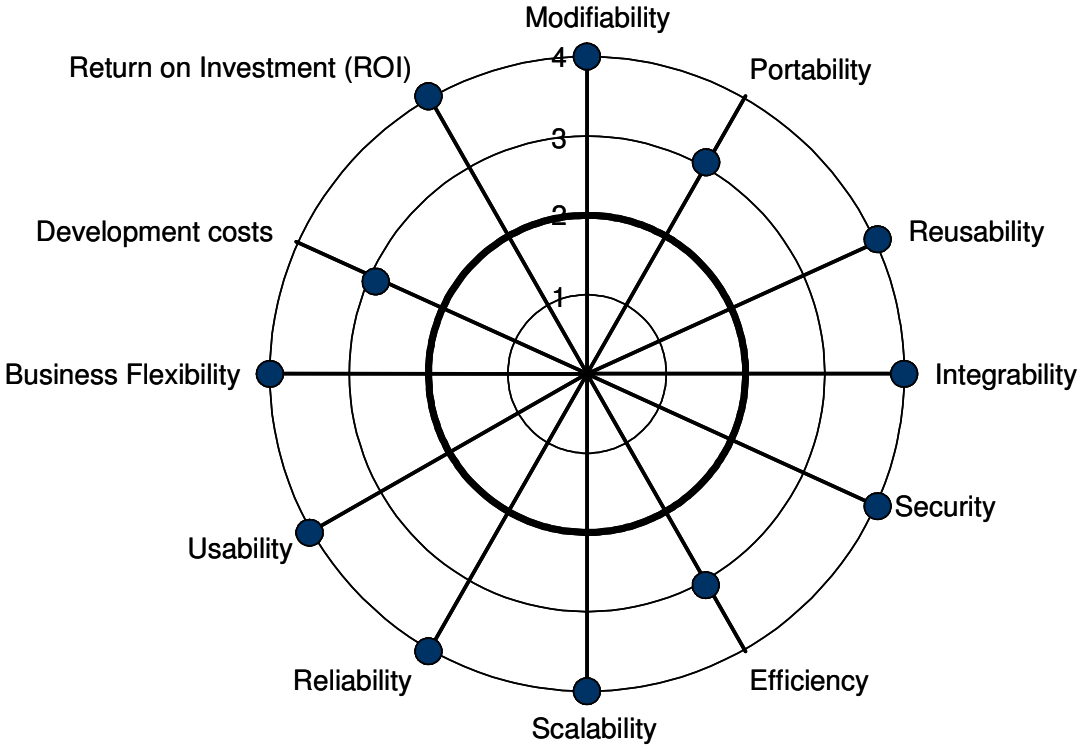
‘As-it-was’ results



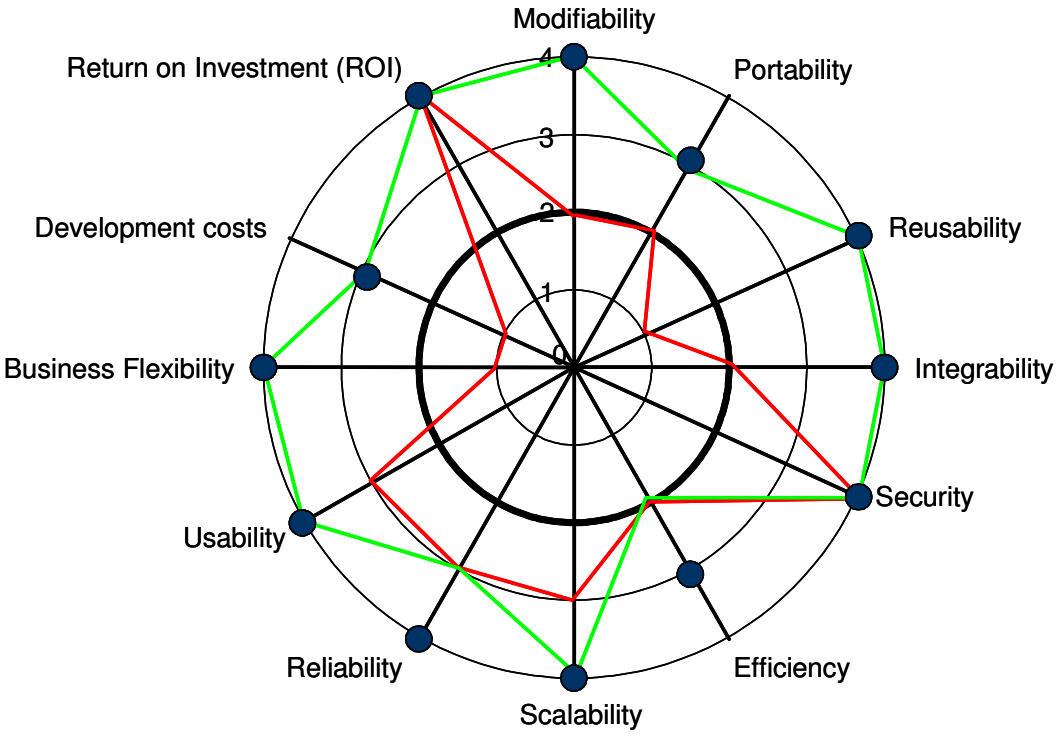
‘As-it-is’ results



Weightening results



Gathered results



LIST OF REFERENCES

HAVING USED BOTH PRUBLISHED MATERIAL, CONFIDENTIAL INFORMATION OF UBS AND IBM, AS WELL AS VERBAL STATEMENTS, THIS CHAPTER HAS BEEN DIVIDED ACCORDINGLY.

PUBLISHED MATERIAL

Architecture & SSP. (May 2005). *Inside SSP*. UBS Corporation (slides): 1-70

Architecture & SSP. (July 2005). *Our new Information systems – Structure and benefits*. UBS Corporation (slides): 1-22

Bass, Len & Clements, Paul & Kazman, Rick. (1998). *Software architecture in practice*. Reading-Massachusetts: Addison-Wesley Longman

Bencher, D. L.. (1994). *Technical Forum – Programming quality improvement in IBM*. IBM Systems Journal 33(1): 215-219
URL http://www.research.ibm.com/journal/sj/331/bencher.pdf, 30.05.2006

Bergmann, Chuck. (2005). *Power your Service Oriented Architecture with an Enterprise Service Bus.* IBM Corporation (slides): 1-53

Bieberstein, Norbert & Bose, Sanjay & Fiammante, Marc & Jones, Keith & Shah, R.. (2006). *Service-oriented Architecture Compass – Business Value, Planning, and Enterprise Roadmap*, Upper Saddle River: Pearson as IBM Press

Boehm, B.W. & Brown, J.R. & Kaspar, H. & Lipow M. & McCleod, G.J., & Merritt, M.J.. (1978). *Characteristics of Software Quality*. Amsterdam: North-Holland

Bratthall, L. & Wohlin, C.. (2000). *Understanding some Software Quality Aspects from Architecture and Design Models*. presented at the 8th IEEE International Workshop on Program Comprehension (IWPC 2000)

Bryman, Alan. (1997). *Kvantitet och kvalitet i samhällsvetensaplig forskning*. Lund: Studentlitteratur

Business & Application Architecture. (2004). *Application Architecture Framework UBS-WMBB – "applikatorische Ziellandschaft"*. UBS Corporation (slides): 1-37

Centre of Software Engineering (Essi-Scope). (2003). *ISO/IEC 9126:1991: The Standard of reference* [www document]. URL http://www.cse.dcu.ie/essiscope/sm2/9126ref.html, 18.05.2006

Chappell, David A.. (2004). *Enterprise Service Bus: Theory in Practice*. USA: O'Reilly Media

Clark, Mike & Fletcher, Peter & Hanson, J. Jeffrey & Irani, Romin & Waterhouse, Mark & Thelin, Joergen. (2002). *Web Services Business Strategies and Architectures*. UK: Wrox Press

Elmasri, Ramez & Navathe, Shamkant B.. (2004). *Fundamentals of Database Systems (4th Edition)*. Boston: Pearson Education

Escher, Thomas K.. (2005). *IT-Plattformwechsel in einer Grossbank*, UBS Corporation (slides): 1-11

Gillspie, Andrew. (1999). *Advanced Economics through Diagrams*. Oxford: Oxford University Press

Gold-Bernstein, Beth. (2004). *Integration Best Practices: Beyond The SOA, Composite Apps And Web Services Hype* [www document]. URL http://www.ebizq.net/hot_topics/esb/features/4894.html?page=2, 10.05.2006

Grady, R. & Caswell, D.. (1987). *Software Metrics: Establishing a Company-Wide Program*. Englewood Cliffs: Prentice-Hall

Green Hills Software Inc. (2006). *Maximize your rate of return on investment* [www document]. URL http://www.ghs.com/MaximizeROI.html, 18.06.2006

Grey, William & Katircioglu, Kaan & Bagchi, Sugato & Shi, Dailun, & Gallego Guillermo & Seybold, Dave & Stefanis, Stavros. (2003). *An Analytic Approach for Quantifying the Value of e-Business Initiatives*. IBM Systems Journal 42(3): 484-497. URL http://researchweb.watson.ibm.com/journal/sj/423/grey.pdf, 01.06.2006

Gruman, Galen. (2006). *SOA Pioneers Goes the Distance*. Info World 28(2): 25-28

Hammerschall, Ulrike. (2005). *Verteilte Systeme und Anwendungen – Architekturkonzepte, Standrads und Middleware-Technologien*. Munich: Pearson Studium

Hasselbring, Wilhelm. (2006). *Software-Architektur.,* Oldenburg: Springer-Verlag. URL http://www.gi-ev.de/service/informatiklexikon/informatiklexikon-detailansicht/meldung/128/, 25.05.2006

Heineman, George T. & Councill, William T.. (2001). *Component-Based Software Engineering – Putting the pieces together*. Upper Saddle River: Addison-Wesley

Huotari, Maija-Leena & Wilson, T.D.. (2001). *Determining organizational information needs: the Critical Success Factors approach* [www document]. Information Research 6(3). URL http://informationr.net/ir/6-3/paper108.html, 22.09.2006

IEEE with Radatz, Jane as chairperson. (1990). *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-90: 1-84. URL http://80-ieeexplore.ieee.org.ludwig.lub.lu.se/iel1/2238/4148/00159342.pdf, 17.03.2006

ISO. (1986). *ISO 8402 Quality Vocabulary*. Geneva: International Organization for Standardization

ISO/IEC. (1994). *Informationstechnik - Bewerten von Softwareprodukten - Qualitätsmerkmale und Leitfaden zu ihrer Verwendung (Identisch mit ISO/IEC 9126:1991).* DIN 66272(1994-10): 1-10

Kan, Stephen H.. (2003). *Metrics and models in Software Quality Engineering (Second Edition)*. Upper Saddle River: Pearson Education

Keen, Martin & Acharya, Amit & Bishop, Susan & Hopkins, Alan & Milinski, Sven & Nott, Chris & Robinson, Rick & Adams, Jonathan & Verschueren, Paul. (2004). *Patterns: Implementing an SOA using an Enterprise Service Bus*. USA: IBM Redbooks
URL http://www.redbooks.ibm.com/redbooks/pdfs/sg246346.pdf, 04.05.2006

Kozaczynski, Wojtek. (2002). *Requirements, Architectures and Risks, Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02)* [www document].
URL http://csdl2.computer.org/comp/proceedings/re/2002/1465/00/14650006.pdf, 03.04.2006

Krcmar, H.. (2000, 2002, 2005). *Informationsmanagement. 4. Aufl.*. Berlin Heidelberg: Springer

Kvale, Steinar. (1997). *Den kvalitativa forskningsintervjun*. Lund: Studentlitteratur

Lager, Marshall. (2006). *SOA simple.* Customer Relationship Management 10(2): 20-24

Lamont, Judith. (2006). *Service-oriented architecture: a way of life*. KM World 15(2): 20-21

Lundahl, Ulf & Skärvad, Per-Hugo. (1982, 1999). *Utredningsmetodik för samhällsvetare och ekonomer*. Lund: Studentlitteratur

Löwgren, Jonas. (1993). *Human-computer interaction – What every system developer should know*. Lund: Studentlitteratur

McCall, J.A. & Richards, P.K. &Walters, G.F.. (1977). *Factors in Software Quality*. Springfield: NTIS

Miles, Matthew B. & Huberman, Michael A.. (1994). *An Expanded Sourcebook – Qualitative Data Analysis*. USA: SAGE Publications

Natis, Y & Schulte, Roy. (2003). *Introduction to Service-Oriented Architecture*. Gartner Research – Research Note SPA-19-5971(April): 1-6

Ortega, Maryoloy & Pérez, María & Rojas, Tersita. (2003). *Construction of a Systemic Quality Model for Evaluating a Software Product*. Software Quality Journal 11(3): 219–242

Palani Rajan, P.K. & Van Wie, M. & Campbell, M. & Otto, K. & Wood, K.. (2003). *Design for flexibility – Measures and Guidelines*. International Conference on Engineering Design ICED 03 in Stockholm

Patel, R. & Davidson, B. (1991). *Forskningsmetodikens grunder. Att planera, genomföra och rapportera en undersökning.* Lund: Studentlitteratur

Peisl, Roland. (2006). *ProcessArt – Geschäftsprozessmanagement trifft Service Orientierte Architekturen BPM und SOA von IBM*, IBM Corporation (slides): 1-23.
URL http://www.it-business.net/konferenz/download/ibm/27032006/ibm_27032006.pdf, 30.05.2006

Plummer, D.. (2002). *SODA helps developers do application integration*. Gartner Research SPA-18-6402(November): 1-5

Porter, Michael E.. (1998). *On competition.* USA: Harvard Business Review Book

Rockart, J.F.. (1979). *Chief executives define their own data needs*. Harvard Business Review, 57 (2): 238-241

Schulte, Roy. (2002). *Predicts 2003: SOA Is Changing Software.* Gartner Research - Article Top View AV-18-9758(December): 1-4

Snee, R.D.. (2004). *Six Sigma: the evolution of 100 years of business improvement methodology*. International Journal of Six Sigma and Competitive Advantage 1(1): 4-20

Szyperski, Clemens. (2002). *Components Software. Beyond Object-Oriented Programming (Second Edition)*. Harlow: Addison-Wesley

Stoyan, Robert. (2004). *Management von Webprojekten – Führung, Projektplan, Vertrag. Mit Beiträgen zu IT, Branding, Webdesign und Recht.* Berlin Heidelberg: Springer

The Standish Group International Inc. (2001). *Extreme chaos*. CHAOS Report:1-12.
URL http://www.standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf, 15.03.2006

The Standish Group International Inc. (2004) *2004 third quarter research report*, CHAOS Report: 1.
URL http://www.standishgroup.com/sample_research/PDFpages/q3-spotlight.pdf, 21.06.2006

Unknown. (2003). *Realigning IT for service delivery*. Middle East Banker 36(June): 1. URL http://www.bankerme.com/bme/2003/jun/it_in_banking_2.asp, 01.06.2006

Unknown. (2003). *A brief introduction to the Abacus* [www document]. URL http://www.ee.ryerson.ca/~elf/abacus/intro.html, 19.04.2006

Wong-Bushby, I. & Egan, R. & Isaacson, C.. (2006). *A case study in SOA and Re-Architecture at Company ABC*. Proceedings of the 39th Annual Hawaii International Conference on System Science

UNPUBLISHED MATERIAL

Arsanjani, Ali (SOA Centre of Excellence). (2006). *Service Integration Maturity Model (SIMM): Concepts*. IBM Corporation (slides): 1-30. (IBM confidential)

Beauchemin, Dr. S. S.. (2003). *CS377 Software Quality Assurance* [www document]. URL http://www.csd.uwo.ca/~beau/CS377/CS377-SQA.html, 09.05.2006

Bieberstein, Norbert & Bose, Sanjay & Fiammante, Marc & Zimmernann, Olaf. (2005). *SOA Roadmap – The EIS SOA Cook Book*. IBM Corporation (pdf) p. 1-196 (IBM confidential)

Blakely, Beth. (2002). *ROI for Web Services: Risk factors*, [www documents]. URL http://www.zdnet.com.au/insight/0,39023731,20270041,00.htm, 16.06.2006

Dahan, Udi. (2004). *Client, server, SOA?* [www document]. URL http://udidahan.weblogs.us/archives/014716.html, 17.05.2006

Ebner, John. (2003). *Strategic Solution Program - The "Big Picture"*. UBS Corporation (slides): 1-242. (UBS confidential)

Glinz, Prof. Dr. Martin. (2001, 2003). *Spezifikation und Entwurf von Software*. Universität Zürich (lecture slides):chapter 1-26

Harishankar, Ray. (2001). *A Global Technology Company's Registration Application Scalability, Performance & Availability Assessment*. IBM Global Services / BIS Architecture Centre of Excellence (document):1-25. (IBM confidential)

IBM Software University. (2006). *Session 3613b – Secure SOA*. IBM Corporation (slides): 1-51 (IBM Confidential)

Known Library. (2004). *UBS Artikel* [www document]. URL http://ubs.know-library.net/, 19.04.2006

Langel, Randy. (2004). Service *Oriented Architecture Business Value Proposition*. IBM Corporation (slides): 1-43. (IBM confidential)

Opplinger, PD Dr. Rolf. (2005). *Sicherheit in der Informationstechnik*. Universität Zürich (lecture slides): chapter 1-5

Reinitz, Rachel. (2003, 2004, 2005). *Web Services Architecture Best Practices for SOA and ESB*. IBM Corporation (slides): 1-138. (IBM confidential)

Roik, Nicole & Balzer, Yvonne. (2004). *Cost Benefit Research for the Implementation of SOA for a financial institute*. IBM Corporation (slides): 1-19. (IBM confidential)

Schmelzer, Ronald. (2005). *The ROI of SOA ZapFlash* [www document]. URL http://www.zapthink.com/report.html?id=ZAPFLASH-20050127, 18.06.2006

Tagesanzeiger. (2006). *Ospel verteidigt Lohnpolitik der UBS* [www document]. URL http://www.tagesanzeiger.ch/dyn/news/wirtschaft/615046.html, 19.04.2006

Turner, K.. (2003). *An Introduction to Value Driver Analysis*. IBM Corporation (slides): 1-118

UBS Business & Application Architecture (translated by Furth, Norbert). (2004). *Renewal of the IT-Landscape Switzerland – Approach and Phases*. UBS Corporation (slides): 1-46

UBS. (1998-2006a). *UBS in a few words* [www document]. URL http://www.ubs.com/1/e/about/ourprofile.html, 11.04.2006

UBS. (1998-2006b). *Organizational Structure – How we are organized* [www document]. URL http://www.ubs.com/1/e/about/ubs_group/group.html, 11.04.2006

UBS. (1998-2006c). *In a few figures* [www document]. URL http://www.ubs.com/1/e/about/keyfigures.html, 11.04.2006

UBS. (1998-2006d). *Our Business* [www document]. URL http://www.ubs.com/1/e/about/our_businesses.html, 10.04.2006

UBS. (1998-2006e). *About us* [www document]. URL http://www.ubs.com/1/e/about.html, 14.07.2006

UBS. (2005). *Our new information systems - Structure and benefits.* UBS Corporation (slides): 1-24. (UBS Confidential)

Weisser, Jason. (2004). *The challenge to integration: Java to .NET and beyond*, [www document]. URL http://www.developers.net/solutions/181, 14.03.2006

Wikipedia. (2006). *Service-oriented architecture* [www document]. URL http://en.wikipedia.org/wiki/Image:SOA_Elements.png, 20.04.2006

Wikipedia. (2006). *Scalability* [www document]. URL http://en.wikipedia.org/wiki/Scalability, 01.06.2006

Wolfe, Martin. (2003-2004). *Enterprise Technology Readiness Model - Current Version: v1.1-b9-*. IBM Corporation (slides): 1-44. (IBM confidential)

Young, Richard & Biz/ed. (1996-2006). *Basic Economic Problems – Notes* [www document]. URL http://www.bized.ac.uk/learn/economics/micro/problem/notes.htm, 03.04.2006

VERBAL EXCHANGE

Furth, Norbert, 2006, IT-Architect, IBM Switzerland

Krüsemann, Gisbert, 2006, IT-Architect, IBM Switzerland

von Bülzingslöwen, Günter, 2006, IT-Architect, UBS Switzerland