UPTEC IT09 001
Examensarbete 30 hp
Januari 2009

SOA

How to Build Winning Enterprises with IT

Mikael Nordström

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

SOA - How to Build Winning Enterprises with IT

Mikael Nordström

This paper shows that Service Oriented Architecture (SOA) systems can be developed at lower costs, less risk and with a faster time to market than a traditional developed IT system. The primary reason for this is that SOA enables reuse; by loosely coupled reusable services and business processes.

However, this is not the main benefit; still a great side effect. SOA can and should deliver good value for the business. The main benefit with SOA is therefore to create new and better opportunities for the business. Today's rapid changes and fierce competition increases the importance of flexible enterprises; SOA has the ability to develop and modify services quickly which leads to flexible and thus winning enterprises.

It is important to keep in mind that SOA is an architecture, not a technology. Like any other architecture one finds value over time; long term thinking is important in the implementation of SOA.

This paper also covers a number of technologies common in SOA, and a prototype to show how these technologies can be implemented in a SOA system.

Handledare: Fredrik Bergman
Ämnesgranskare: Mats Daniels
Examinator: Anders Jansson
ISSN: 1401-5749, UPTEC IT09 001
Sponsor: EnjoyIT

Tryckt av: Reprocentralen ITC

Sammanfattning

Traditionellt utvecklade distribuerade IT-system har på grund av vidareutveckling och underhållsarbete blivit mer och mer komplexa med tiden. Detta har lett till att systemen blivit väldigt komplicerade och därmed dyra att underhålla och vidareutveckla. Komplexiteten i systemen innebär även att affärsverksamhetsutvecklingen hindras, vilket inte är bra för affärsverksamheten. Tjänsteorienterad arkitektur, eller *Service Oriented Architecture* (SOA) som den engelska termen är, har på senare år dykt upp som en affärscentrerad IT arkitektonisk strategi för att bygga just IT-system efter affärsverksamhetens affärsprocesser och krav.

IT-chefer i Sverige anser att de största hindren för att genomföra IT-projekt är kostnader, risker och tid. Detta leder till frågan om det med SOA går att bygga IT-system med lägre kostnader, mindre risker och snabbare resultat än traditionella utvecklingsmetoder. Studien visar att detta är möjligt att uppnå med SOA.

Den största anledningen till att dessa hinder, kostnader, risker och tid, kan minimeras är att SOA byggs med flexibla, återanvändbara tjänster och affärsprocesser. Dessutom främjar SOA återanvändningen av befintliga system vilket innebär att dessa system inte behöver bytas ut direkt, även om visionen kan vara att byta ut dem i framtiden. På lång sikt kan de totala IT-kostnaderna minskas med 20 % enligt Gartner.

Detta är trots allt inte den största fördelen med SOA, dock en riktigt bra sidoeffekt. SOA kan, och bör leverera värde för verksamheten. Den största fördelen med SOA är därför att skapa nya och bättre möjligheter för företagen. Med tanke på de snabba förändringarna och den hårda konkurrensen som råder i världen idag måste framgångsrika företag skapa en flexibel organisation som snabbt kan rätta sig efter samhället och kundernas behov. SOA har den förmågan; att utveckla och ändra tjänster och affärsprocesser snabbt vilket leder till flexibla, och därmed framgångsrika företag.

Det finns några viktiga saker att tänka på vid införandet av SOA. Ett lyckat SOA-projekt behöver stöd från toppen av företaget, vilket är nödvändigt för att förvekliga en förändring i organisationen. En annan viktig sak att ha i åtanke är att SOA är en arkitektur, inte en teknologi. Liksom andra arkitekturer är det viktigt att tänka långsiktigt; det kanske är först vid andra eller till och med tredje projektet som de stora fördelarna med SOA upptäcks.

Denna rapport går dessutom igenom ett antal teknologier som är vanliga inom SOA samt visar med en utvecklad prototyp hur dessa teknologier kan användas i ett SOA-system. Prototypen visar på att flera av fördelarna med SOA redan infinner sig vid ett litet system.

Table of contents

1 Introduction

1.1 Background

There are several things that make an enterprise successful; according to Bedredinov & Bedredinov (Bedredinov & Bedredinov, 2008) *"It is widely known that the efficiency of its business processes is a passport to success for any company"*. Since most business processes nowadays use IT, it means that IT is a very important part of a successful enterprise.

Over years, IT systems have become more and more complex with tightly coupled connections that make systems difficult to maintain and develop. This has led to business process fallen into the hands of IT, which is not good for business. What is not good for the business is ultimately not good for IT because without the business, IT ceases to exist. (Hurwitz, Bloor & Baroudi, 2006) (Bagwell, 2007)

As reflected in Figure 1, applications in traditional developed systems are siloed and closed, and it is not uncommon with duplication of code and information.

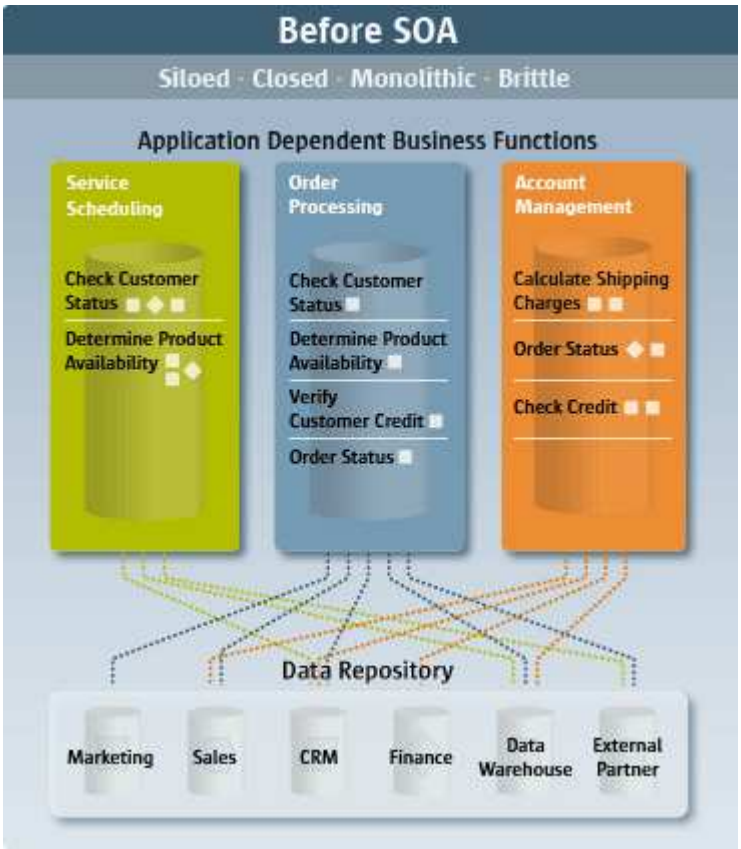

Figure 1. IT systems before SOA (Sun SOA, 2008)

Years have passed and enterprises started to think how they could build IT systems based on their business processes and not the other way round. Attempts have been made, including Information Engineering and Object Oriented Programming with decent, but not the results hoped for. A few years ago Service Oriented Architecture (SOA) showed up and enterprises were said that they now could align IT with business. This was a major breakthrough that made it possible to build IT systems out of their business processes and not the contrary. From now on, Service Oriented Architecture will be called SOA. (Hurwitz, Bloor & Baroudi, 2006)

It is often a problem that enterprises have old IT systems that fulfill their tasks very well; at the same time it is very difficult and expensive to further develop these systems. According to a study by IT-konsulten Unisys (Thurfjell, 2008) the main obstacles for IT projects are:

1. **Cost** - Although IT modernization will lead to reduced costs or increased profits in the future, IT managers feel that it is difficult to get funding to implement changes.

2. **Risks** - To change the business critical applications is a great risk; if it does not work as it should it means big problems.

3. **Slowness** – It can be difficult to get through a change project if everything flows on as it should.

4. **Time** – Complex modernization projects requires planning and time.

According to Bedrinov and Bedrinov enterprises need a good business idea and an efficient business process to be successful. But the development of IT systems for these business processes cost money, it takes time and it is not risk free. These things, money, time and risks, are all obstacles for an IT project to succeed. (Bedredinov & Bedredinov, 2008) (Thurfjell, 2008)

1.2 Purpose

The purpose of this paper is to show whether and why successful enterprises should build their business systems and business processes with SOA instead of other traditional development practices.

1.3 Problem statement

No enterprises are making money on the implementation of an IT system; it is the underlying business idea and process that makes profit. But if enterprises can keep development costs low and minimize risks, it indirectly means faster and safer profit.

Is it possible to develop an IT system with the help of SOA by which development costs are kept low, the risks are minimized and where it is possible to see results within a short time of period? If this is possible, it opens the possibility for the implementation of several IT projects in a successful manner and may therefore make money on IT with the help of SOA.

1.4 Scope

SOA covers a wide spectrum and includes a lot from business idea and processes down to IT and the implementation of IT. Since it would take several months and pages to cover the whole area of SOA, this thesis focuses on IT and especially the implementation of IT in an SOA system.

1.5 Method

1.5.1 Literature study

A literature study has been carried out, mostly by reading articles from Internet but also a couple of books in the field of SOA. This has been a large part of the work which has led to the discussion section at the end of this document. Beside the theoretical part, technologies to develop an SOA system have been studied. This has mostly been made with materials from IBM, like IBM Redbooks and developerworks, which is a site for IBM developers. This has led to a prototype of an SOA system.

1.5.2 Prototyping

A prototype of the implementation part of an SOA system has been developed. It is developed mostly with IBM products and displays on a small scale how an SOA system can be built.

1.6 Outline

Next chapter is about SOA, its foundation and comprehensive information about SOA and how it works. This chapter is followed by a number of chapters with selected essential

building blocks of an SOA system. These building blocks are actually a number of technologies considered, but need not necessarily belong to an SOA system. This is followed by a section about the prototype that has been developed; to show how it is to implement these technologies for real. The report concludes with a discussion about the results and other interesting things that appeared in the meantime.

1.7 Readers

This report is directed to those of you who are interested in IT, SOA, or implementation of business systems and processes with IT systems.

2 SOA – Service Oriented Architecture

SOA is not another product to build IT systems; it is a concept for how to build flexible IT systems with existing technologies. There are about as many definitions of SOA as there are people who know what it is. The book Service Oriented Architecture For Dummies (Hurwitz, Bloor & Baroudi, 2006) defines SOA as *"a software architecture for building applications that implement business processes or services using a set of loosely coupled black-box components"*, while IBM defines SOA as *"a business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks, or services"*. (IBM, 2008)

> *"Rapid change, fierce competition and an ever-flattening world economy are driving the need for superior business agility. A new class of truly agile organizations, the globally integrated enterprise, is emerging as the winner. How? By delivering unique value, tapping into the power of globalization and forging a strategy of componentization.*
>
> *These organizations understand that using service-oriented architecture (SOA) is a preferred method of delivering sustainable agility. They need this agility - that is, the ability to quickly and effectively respond to changes, opportunities and threats - to compete effectively." (IBM – Smart SOA Approach, 2008)*

According to Chris Harding (Harding, 2007) the key differences between SOA and other architectural styles are:

- **Loosely coupled** – The modules are loosely coupled services that can be combined dynamically, as opposed to subroutines, scripts, or other forms of program that invoke each other directly.

- **Emphasis on infrastructure support** – There is more emphasis on infrastructure support for service development and evolution. The enterprise architect has always been concerned with principles and guidelines governing the evolution of the architecture components over time. With SOA, this extends to the specification of

particular development tools and methods, and their incorporation in the infrastructure to provide software service lifecycle support.

- **Rapid changes** – The ability to develop and modify services rapidly, the need to ensure that they support the business operations as effectively as possible, and the desire to encourage their reuse by different parts of the enterprise impacts on governance: the process by which the translation of the architect's specifications into implemented systems, and the continuing evolution of those systems, is controlled.

- **Available information** – Information is not locked up in specific services, as it often is in the so called "silo" applications of earlier architecture styles, but is available when and where needed throughout the enterprise and its partners, unhindered by artificial boundaries imposed by the enterprise's information technology.

Figure 2 shows the same system as in Figure 1, but now built with SOA. Services (functions) are loosely coupled and reusable which leads to a more flexible system where applications now can “talk to each other”.

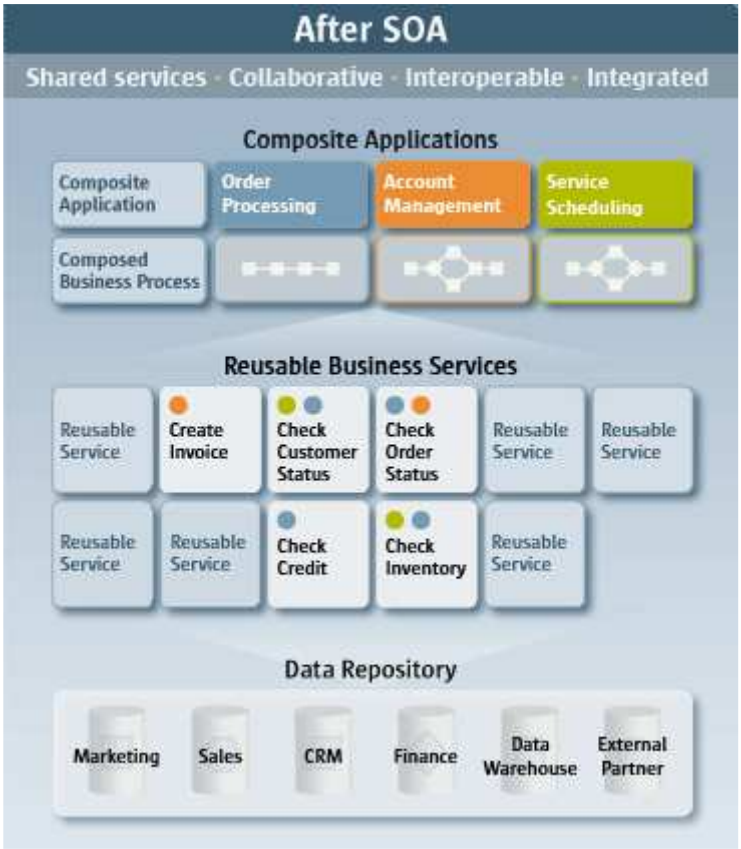

Figure 2. IT systems after SOA (Sun SOA, 2008)

Today's financial crisis in Sweden, and large parts of the world, has struck hard against the financial sector, especially banks. According to Magnus Höij, chief editor of Internet World, banks would be able to cope with rapid business changes more quickly with an SOA system than with today's *"spaghetti systems"*. He argues that today's banks are totally dependent on other banks and financial operators in other markets; if any actor tumbles, it affects everyone else in the system. (Höij, 2008)

2.1 Aligning IT with Business

Over years IT systems has been developed into tightly coupled, complex systems. This has led to business requirements to comply with IT solutions, which means that business development has been limited because of IT. This is not good for business, and what is not good for business is not good for IT. Enterprises want to do *vice versa* and align IT with their business goals; SOA enables this. (Hurwitz, Bloor & Baroudi, 2006)

> *"To succeed, SOA must be sponsored by top management. But committing technical resources is not enough. Management must understand how SOA relates to the enterprise as a whole, and be prepared if necessary to change its culture, to gain full benefit from SOA." (Harding, 2007)*

2.2 Industry standards

IT architects and developers have under a long time tried to reuse parts of IT systems to other IT systems. This has been very hard to perform until recent years. Internet and other industry standards have enabled the implementation of SOA systems. This has led IT development to a new dimension which means that the wheel does not need to be reinvented for each new system anymore. (Hurwitz, Bloor & Baroudi, 2006)

One of those key industry standards developed in recent years is XML, which is the basis for many of the standards used by SOA, like the transport protocol SOAP. These and other important standards used by SOA will be addressed later in this document.

2.3 Reusability

Today's businesses are changing continuously with the result that IT systems always need to be developed to line up with the businesses. SOA does not mean that you take out the old systems and replacing with a new one. Instead SOA enables the reuse of old systems, called legacy systems, and to develop the new parts out of the legacy systems as reusable, loosely coupled black boxes. (Bagwell, 2007)

> *"In a service-oriented architecture world, business applications are assembled using a set of building blocks known as components – some of which may be available "off the shelf," and some of which may have to be built from scratch." (Hurwitz, Bloor & Baroudi, 2006)*

A component, or service, have a clear interface that specifies what the service does, what the input should be and what output it responds with. You do not have to know what hides behind the interface, if it invokes other systems or databases. The only interesting thing is what the service performs. Such a service can then be reused by multiple systems in an enterprise, or even of systems outside the company, for example by partners, if allowed. These services can be used by reusable business components or be presented for users by some sort of user interface.

There are currently a number of standards for developing a service in SOA; the most common, as almost have become associated with SOA, is Web Services. Web Services will be presented in more detail in later chapters. By implement SOA reuse through service creation will realize following value. (IBM – developerworks, 2008)

- **Less expensive** – It is less expensive to reuse existing applications than to write new applications from scratch.

- **Lower risks and faster time to market** – Reusing proven and time-tested applications results in lower risks and faster time to market.

- **Less maintenance** – Maintenance overhead shrinks with greater use of proven and tested codes for common functions.

2.3.1 Encapsulated

As mentioned earlier a service is like a small black box. What is important is the type of behavior a service provides, not how it is implemented. A Web Service Description Language (WSDL) document is the mechanism how to describe the behavior encapsulated by a service. WSDL documents are defined in later chapters. Some key values for encapsulation, defined by IBM, are listed below. (IBM – developerworks, 2008)

- **Coping with complexity** – System complexity is reduced when application designers do not have to worry about implementation details of the services they are invoking.

- **Flexibility and scalability** – Substitution of different implementation of the same type of service, or multiple equivalent services, is possible at runtime.

- **Extensibility** – Behavior is encapsulated and extended by providing new services with similar service descriptions.

2.3.2 Loosely coupled

SOA provide built-in mechanisms to facilitate loose couplings between services and other components of an application. SOA ensures that the service is decoupled from other components in location, protocol, and time. (Hanson, 2002)

Services must be able to be located anywhere and then be discovered and used by other components or applications dynamically. SOA provides a mechanism for location transparency using service registration. Services are also protocol independent. That is, they operate in the same manner regardless of what protocol was used to communicate with them. The communication protocol support is provided by the SOA platform. Services may be called synchronously or asynchronously. (Hanson, 2002)

2.3.3 Legacy systems

SOA allows development of secure services with high quality. SOA can also extend and protect enterprises current IT systems, called legacy systems. By transforming legacy applications into flexible services, the reuse of well-tested applications reduces development costs and minimizes risks. This means that not everything needs to be replaced immediately;

the old systems can be used together with an SOA solution represented as services. (IBM – developerworks, 2008)

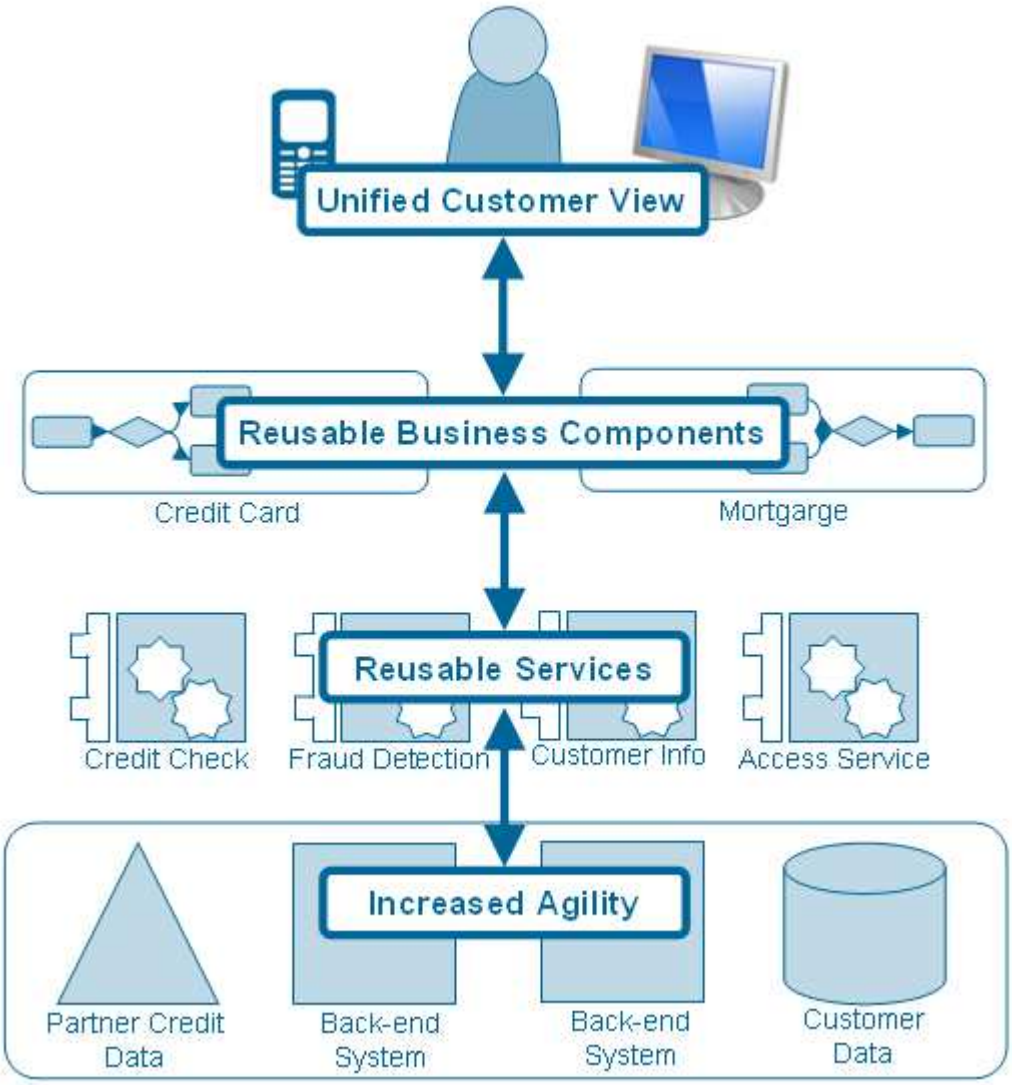

Figure 3. SOA Overview, shows the different layers within SOA.

2.4 Build winning enterprise with SOA

Gartner predicts that SOA development reduces overall IT costs in the long term by 20% compared with traditional methods. The savings grow exponentially over time as business services develop and greater reuse is achieved. But just looking at cost savings, the opportunities for dramatic business process improvements that SOA can deliver will be missed. (Roch, 2006)

When The SOA Consortium and CIO Magazine jointly appointed the winners of a major SOA contest they could find several common denominators among the winners. These eight things that characterize a winning SOA effort are treated below. (Röhne, 2008)

2.4.1 Strong support

A winning project needs to have a strong support from senior members of the management. This is essential to realize a change in an organization. Without support from the top, it is extremely difficult to implement SOA initiatives in a good way. It is also essential to have one or more SOA enthusiasts who run the SOA effort. This may be a risk when the research shows that companies sometimes go back to old methods when SOA firebrands are leaving the company. (Röhne, 2008)

2.4.2 Education

A successful project will generate considerably more value for the business. In some cases, a SOA effort can give a ROI[1] of several billion dollars in a few years. For SOA to really generate value, it is important to understand the business benefits of SOA. It is not about to talk with the business about technology. Perhaps there is no need to even mention the concept of SOA. Instead, business must understand the commercial drivers that can be solved, as faster access to information, integration with customers and partners, and so on. They must simply understand the problems that IT now can solve. (Röhne, 2008)

2.4.3 Centre of excellence

A center of excellence, responsible for the SOA initiative is important for a successful project. Maybe there already exists an Enterprise Architecture with IT governance in place; then only adjustments for SOA effort have to be made. Others who do not have formal IT governance can create one in the context of SOA effort. Companies that have done successful SOA initiatives believe that management was an important success. (Röhne, 2008)

2.4.4 Think big, start small

Start to find interesting services by establishing a well-defined business process. Perhaps there is a business process already in place, otherwise a business process can be changed and made

[1] In finance, Return On Investment, ROI, is the ratio of money gained or lost on an investment relative the amount of money invested.

more effective to allow for new services. Always start with one business process at time, instead of starting with all processes simultaneously. (Röhne, 2008)

2.4.5 Define your services

Spend much time on what services that can be critical for the core business, and precise limits for each service. Develop relatively large services with multiple functions rather than a service for each function. Successful SOA efforts have developed quite a few critical business services rather than large amount small services. (Röhne, 2008)

2.4.6 Quality Assurance

SOA creates all sorts of new challenges when it comes to the quality of services. Winning SOA projects introduce good practices for quality assurance of services. An example might be to load test each service before it is deployed. (Röhne, 2008)

2.4.7 Think long term

SOA is not a technology, it is architecture. Like any other architecture, you find value over time. It is often during the second, or perhaps third, SOA project that companies discover the great benefits of their SOA effort. (Röhne, 2008)

2.4.8 Deliver business benefits

Successful SOA projects deliver a good value for the business. The idea of SOA is not to fix the IT infrastructure, or reduce development costs; this is sometimes great side effects. The major force behind an SOA effort should be to create new and better opportunities for the business. (Röhne, 2008)

3 Web Services

A Web Service is a software interface that describes application functionality that could be invoked over a network, not necessarily over Internet. Web Services are based on XML to provide an independent platform and programming language environment. (IBM – developerworks, 2008)

Over the last couple of years, Web Services have expanded to become more popular with application developers. In SOA, Web Services have almost become a standard for the implementation of services, even though there are other ways to do it. Web Services technology represents an important way for businesses to communicate with each other and with clients as well. Unlike traditional client/service models, such as a Web Server or Web page system, Web Services do not provide the user with a GUI[2]. Instead, Web Services share business logic, data and processes through a programmatic interface across a network. The applications interface with each other, not with the users. Developers can then add the Web Service to a GUI, such as a Web page or an executable program, to offer specific functionality to users. (Beal, 2005)

> *"The term Web services describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available. Used primarily as a means for businesses to communicate with each other and with clients, Web services allow organizations to communicate data without intimate knowledge of each other's IT systems behind the firewall." (Webopedia, 2008)*

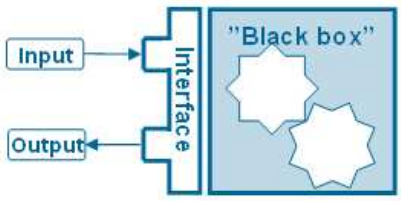

Figure 4. Web Service

[2] A Graphical User Interface, GUI, is a type of user interface which allows people to interact through graphical interfaces.

3.1 XML – eXtensible Markup Language

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML[3]. Originally designed to meet the challenges of large scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. (W3C – XML, 2008)

XML is a set of rules for how textual format should be constructed, the format that lets you organize your data. XML is not a programming language, and you need not be a programmer to use it or learn it. XML makes it easy for a computer to generate data, read data and ensure that the data structures are unambiguous. XML avoid common pitfalls in language design. It is extending scalable, platform independent and it gives the opportunity to express themselves as well as independent specific to place, culture and language. (W3C – XML, 2001)

```
<ALL_CUSTOMER>
  <CUSTOMER>
    <ID>1031</ID>
    <NAME>Uppsala Universitet</NAME>
    <VISITING_ADDRESS>Uppsalavägen 1</VISITING_ADDRESS>
    <ZIPCODE>752 38</ZIPCODE>
    <CITY>Uppsala</CITY>
    <TELEPHONE>018-555 10 10</TELEPHONE>
    <REFERENCE>Carl von Linné</REFERENCE>
    <ORGANISATION_NUMBER></ORGANISATION_NUMBER>
  </CUSTOMER>
  <CUSTOMER>
    <ID>1032</ID>
    <NAME>Enterprise AB</NAME>
    <VISITING_ADDRESS>Jungle</VISITING_ADDRESS>
    <ZIPCODE>555 55</ZIPCODE>
    <CITY>The World</CITY>
    <TELEPHONE>01-100 55 55</TELEPHONE>
    <REFERENCE>Mr Jungle Man</REFERENCE>
    <ORGANISATION_NUMBER></ORGANISATION_NUMBER>
  </CUSTOMER>
  ...
</ALL_CUSTOMER>
```

Example 1. XML example code.

The development of XML started in 1996 and became a W3C Recommendation in February 1998. This might have one believe that it is an immature technology. But in fact the technology is not new. Before XML there was SGML, developed during the first half of the

[3] The Standard Generalized Markup Language, SGML, is a standard metalanguage to define markup languages for documents.

1980s, standardized by ISO 1986, and used by large documentation project. The development of HTML began in 1990. XML's developers chose, based on experience with HTML, simply the best parts of SGML, and the result was as powerful as SGML, and much better structured and easier to use. (W3C – XML, 2001)

> *"The realization of SOA through Web services is intrinsically driven by core XML technologies. The emergence of service-oriented design principles, however, is affecting how XML technologies are utilized and positioned within contemporary solutions." (Erl, 2005)*

3.2 SOAP

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application defined data types, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols. (W3C – SOAP, 2000)

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:tns="http://www.example.org/SPCS/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <tns:AddCustomer>
      <Customer>
        <Id>1031</Id>
        <Name>Uppsala Universitet</Name>
        <VisitingAddress>Uppsalavägen 1</VisitingAddress>
        <ZipCode>752 38</ZipCode>
        <City>Uppsala</City>
        <Telephone>018-555 10 10</Telephone>
        <Reference>Carl von Linné</Reference>
        <OrganisationNumber></OrganisationNumber>
      </Customer>
    </tns:AddCustomer>
  </soapenv:Body>
</soapenv:Envelope>
```

Example 2. SOAP example code.

A SOAP Web Service is the most common and marketed form of Web Service in the industry. Some people simply collapse Web Service into SOAP and WSDL services. SOAP provides a message construct that can be exchanged over a variety of underlying protocols. In other words, SOAP acts like an envelope that carries its contents. One advantage of SOAP is that it allows rich message exchange patterns ranging from traditional request-and-response to broadcasting and sophisticated message correlations. (He 2003)

As curiosity, it could be mentioned that SOAP previously was an acronym for Simple Object Access Protocol. This is not longer the case. (W3C – SOAP, 2007)

3.3 WSDL – Web Service Description Language

Web Service Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document oriented or procedure oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints; services. WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate. (W3C – WSDL, 2001)

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions ...>
  <wsdl:types>
    <xsd:schema targetNamespace="..." xmlns:xsd="...">
      <xsd:element name="AddCustomer">
        ...
      </xsd:element>
      ...
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="AddCustomerRequest">
    <wsdl:part element="tns:AddCustomer" name="parameters"/>
  </wsdl:message>
  ...
  <wsdl:portType name="...">
    <wsdl:operation name="AddCustomer">
      <wsdl:input message="tns:AddCustomerRequest"/>
      <wsdl:output message="tns:AddCustomerResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="..." type="...">
    <soap:binding style="document" transport="..."/>
```

```
    <wsdl:operation name="AddCustomer">
      <soap:operation soapAction="..."/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="...">
    <wsdl:port binding="..." name="...">
      <soap:address location="http://localhost/"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Example 3. WSDL example code.

The purpose of WSDL is to describe a particular service and how this service is built. WSDL is an XML schema that formally defines the framework that describes a service interface; a standard way to represent data types of messages, the operations that are necessary to carry out the message and how the message should be mapped to the network for further transport, see Figure 5. (SOA & WS, 2008)

An application that wants to use a service can read the service WSDL document and thus obtain information about which functions can be called and how. With this information, SOAP messages are created and sent by the sender and translated by the recipient. (SOA & WS, 2008)

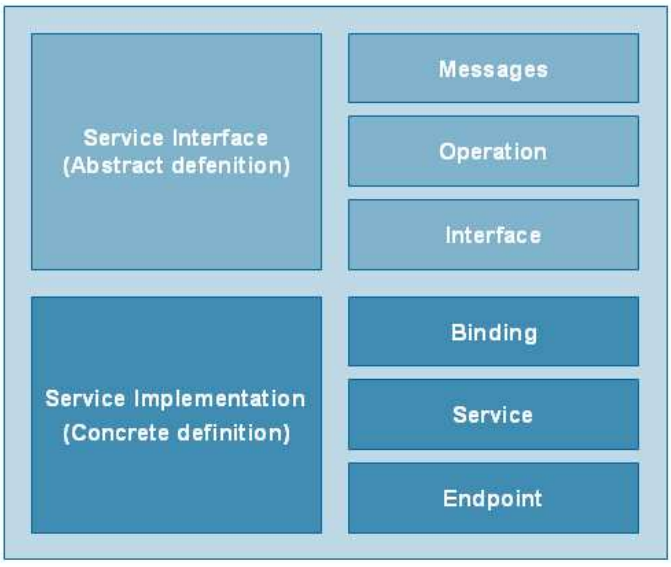

Figure 5. WSDL Conceptual Model

4 ESB – Enterprise Service Bus

An Enterprise Service Bus (ESB) is software infrastructure that simplifies the integration and flexible reuse of business components within a SOA. An ESB provides a dependable and scalable infrastructure that connects disparate applications and IT resources, mediates their incompatibilities, orchestrates their interactions, and makes them broadly available as services for additional uses. (Sonic, 2008)

> *"To integrate old and new, SOA needs an infrastructure that can connect any IT resource, whatever its technology or wherever it is deployed. To be flexible, it needs an infrastructure that can easily combine and re-assemble services to meet changing requirements without disruption. And to be dependable, it needs an infrastructure that is robust and secure. This infrastructure is the ESB." (Sonic, 2008)*

The ESB is not a new software product, but a new way to integrate applications, coordinate resources, and manipulate information. Unlike many previous approaches for connecting distributed applications, such as RPC[4] or distributed objects, the ESB pattern enables the connection of software that runs parallel on different platforms, is written in different programming languages, and uses different programming models. (IBM – developerworks, 2008)

Figure 6. Enterprise Service Bus – connects disparate resources.

[4] Remote Procedure Call, RPC, is a technology that allows applications to cause a subroutine or procedure to execute in other applications.

There are a lot benefits you can get out of an ESB, both for IT and for businesses. Some of the major IT benefits are that an ESB reduces time and effort to integrate new and existing applications it increases flexibility to change complex system behavior by minimizing the hidden dependencies among applications and services in a distributed environment. Another thing is that it reliably delivers messages across services, even after software, network or hardware failure and also disseminates information throughout the enterprise, as well as to customers and trading partners. (Sonic, 2008)

Some of the more interesting benefits for the business are that an ESB reduces the IT costs and improves responsiveness to change business processes quickly and effectively. Beyond this it also leads to reduced cycle times, reduced operating expenses, improved customer service, facilitated mergers and acquisitions and easier and more accurate decision making based on up-to-date business information for "single version of the truth." (Sonic, 2008)

Different vendors have different ideas about what an ESB should contain; most agree that an ESB should manage the following things. (Bagwell, 2007)

- **Messaging services** – Support different message types; content based routing; guarantee message delivery.

- **Management services** – Monitor performance; enforce Service Level Agreement, SLA

- **Interface services** – Support Web Services standards and provide adapters for non-standard interfaces.

- **Mediation services** – Transform messages between different formats.

- **Security services** – Encryption, authentication, authorization.

4.1 SLA – Service Level Agreement

A Service Level Agreement (SLA) is a document which defines the relationship between two parties; the provider and the recipient. It is now widely accepted that service provision and receipt should be governed by an agreement. This is essential to define the parameters of the

service, for the benefit of both the provider and the recipient. This is an important item of documentation for both parties, including performance and security. Service Level Agreements can be handled in SOA Registry. (SLA Zone, 2008) (Service Level Agreement, 2008)

5 SOA Registry

SOA Registry acts like yellow pages for Web Services. It is a resource that provides controlled access to data necessary for governance of SOA projects. In effect, it is a constantly evolving catalog of information about the available services in an SOA implementation. An SOA Registry allows businesses to efficiently discover and communicate with each other using Web Services. (SearchSOA, 2007)

By allowing Web Services to present their WSDL for the SOA Registry, users can dynamically look up what Web Services does and how they should be called, see Figure 7.

An SOA Registry supports the UDDI specification, an XML based registry that was developed for the purpose of making systems interoperable for e-commerce. The SOA registry expands on UDDI by facilitating enhanced and continuous revision of content. Such content can exist as XML documents, process descriptions and information about potential business partners. An SOA Registry allows a participating enterprise to discover and use current and relevant information more quickly than is possible with UDDI alone. (SearchSOA, 2007)

5.1 UDDI – Universal Description Discovery and Integration

A UDDI registry service is a Web Service that manages information about service providers, service implementations and service metadata. Service providers can use UDDI to advertise the services they offer. Service consumers can use UDDI to discover services that suit their requirements and to obtain the service metadata needed to consume those services. (XML Coverpages, 2008)

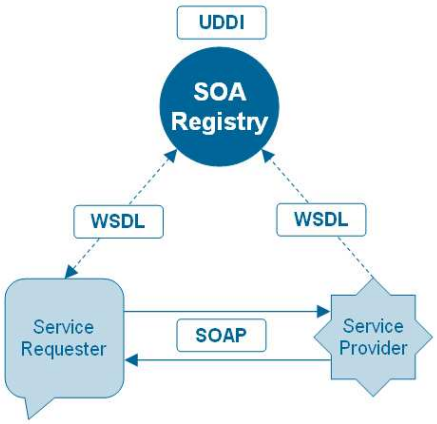

Figure 7. Web Services architecture

6 Business Processes

A business process is as a set of interrelated tasks linked to an activity that spans functional boundaries. Business processes have starting points and ending points, and they are repeatable. Useful business processes make and save money for the enterprise. (IBM – developerworks, 2008)

> *"A business process is a set of linked steps or activities that taken together result in a specific business outcome, either internal or external to the organization." (Microsoft SOA, 2008)*

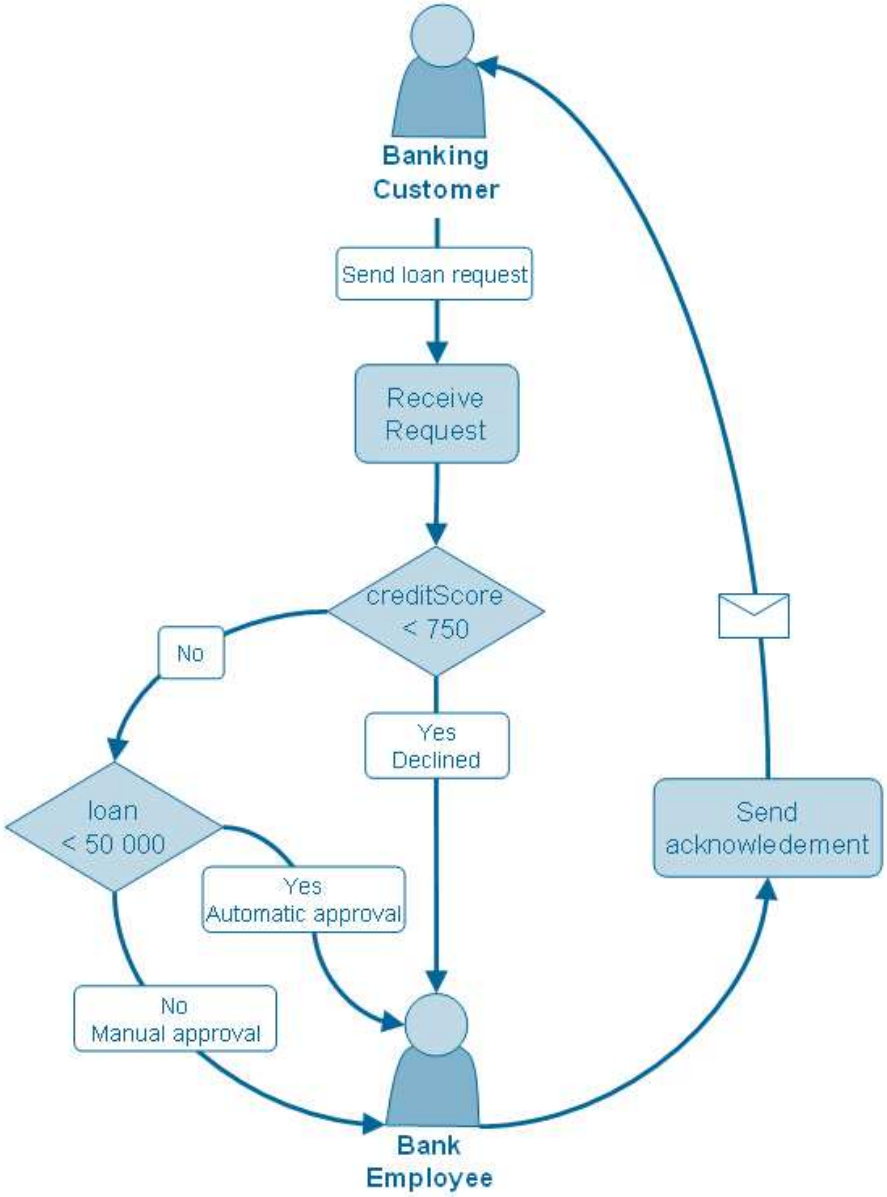

Figure 8. Business Process Sample, Bank loan application

The value of documenting business processes for an enterprise is in the intellectual assets that those processes represent. The widgets that a business produces have value, of course; additionally, the knowledge of how to make those widgets has value too. That knowledge can be captured, added to, and improved in a business process. The analyst gathers information about the current business state; called an "as-is" model. This model is improved to create a future business state, known as the "to-be" model. (IBM – developerworks, 2008)

6.1 BPM – Business Process Management

Business Process Management (BPM) is an important part of SOA; that is an activity performed by an analyst to find out a business current state and the business future roadmap with focus on costs, revenues and timeline bottlenecks. (IMB – developerworks, 2008)

Figure 9 shows the cycle used to develop and streamline business processes continuously.

> *"BPM is a management discipline that combines a process-centric and cross-functional approach to improving how organizations achieve their business goals. Business processes underlie all organizational efforts, and the effectiveness with which they are carried out contributes directly to critical business goals such as customer retention, length of time it takes to fulfill a product order or service, or regulatory compliance. A BPM solution provides the tools that help make these processes explicit, as well as the functionality to help business managers control and change both manual and automated workflows." (Microsoft SOA, 2008)*

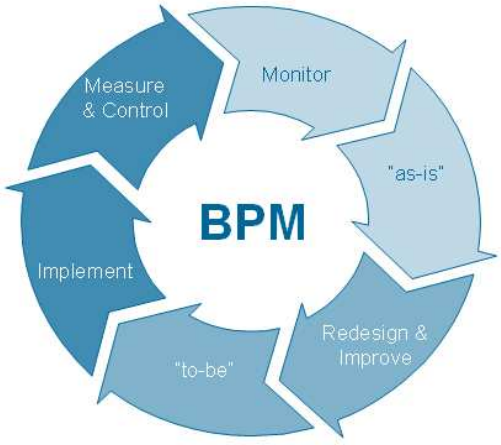

Figure 9. Business Process Management life cycle

7 Prototype – How it works in practice

To get an idea about how it works to implement a small SOA system, with some of the elements mentioned in this report, a prototype has been developed. This is not at all a full fledged SOA system; it is an insight into the implementation of some basic components. The prototype can be seen as the start of a shift from traditional IT systems to SOA architecture with a portal as user interface.

7.1 Development Environments

The prototype is developed with a few different development environments covered below.

7.1.1 Eclipse Platform

Eclipse is a platform composed of extensible application frameworks, tools and runtime libraries for software development and management. Eclipse employs plug-ins to provide all its functionality on top of runtime systems. IBM's strategy is to implement all their middleware products in this framework.

7.1.2 WMB – WebSphere Message Broker

WebSphere Message Broker is an advanced IBM tool that provides transformation between different standard, and non-standard based applications and services. WMB is based on the Eclipse platform and manages transformation and routing of messages.

The Enterprise Service Bus and Web Services in this prototype are developed with WebSphere Message Broker.

7.1.3 Java IDE for Java EE Developers

The Eclipse IDE for Java EE Developers contains everything you need to build Java and Java Enterprise Edition (Java EE) applications. This is also based on the Eclipse platform.

In this prototype, the external web pages are developed with Eclipse IDE for Java EE developers.

7.2 Current Architecture

The current architecture consists of three separate systems; financial system (SPCS), customer relation system (CRM) and time reporting system (TRAP), see Figure 10.

The problem with this architecture is that that the user must log on three different systems and keep track of what applies were. In addition, there is duplicate information in multiple systems, for example, there are customer data in all three systems. Users have also complained that the systems can not cope with to many users at the same time, and that the systems are slow.

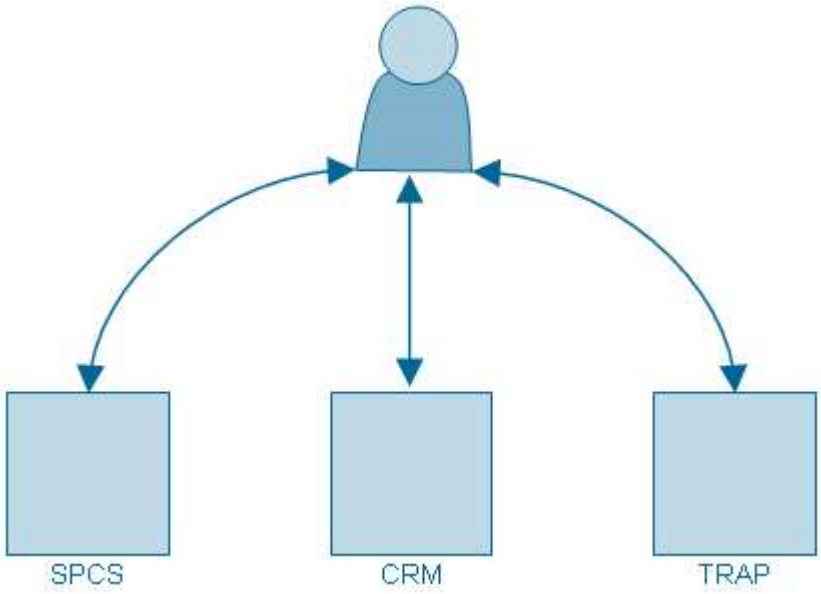

Figure 10. Current Architecture

7.3 Future Architecture

The vision is to move to an SOA solution, which allows users to integrate with the current systems, legacy systems, through a portal via the web. The current systems are presented as Web Services and a new master database for customer data is developed, see Figure 11.

The portal will enable the user to interact with the current systems through a unified user interface. The user need not know which current system he integrates with. This makes it easier for users and can hopefully make users more effective.

The current systems and the master database for customer data will be linked to an ESB that will be developed. This means that the Web Services being developed and linked to the ESB

can integrate with several current systems and customer databases which opens up for new opportunities. The ESB allow that the current systems can integrate with each other.

All existing systems have similar customer data stored. The new customer database will gather all this information, and perhaps more information needed, in the same place. This means that the user integration involving customers will be speeded up when the current, slower, systems need not be called.

Another advantage of this SOA solution is that the enterprise quickly can change the system to the users' and customers' needs.

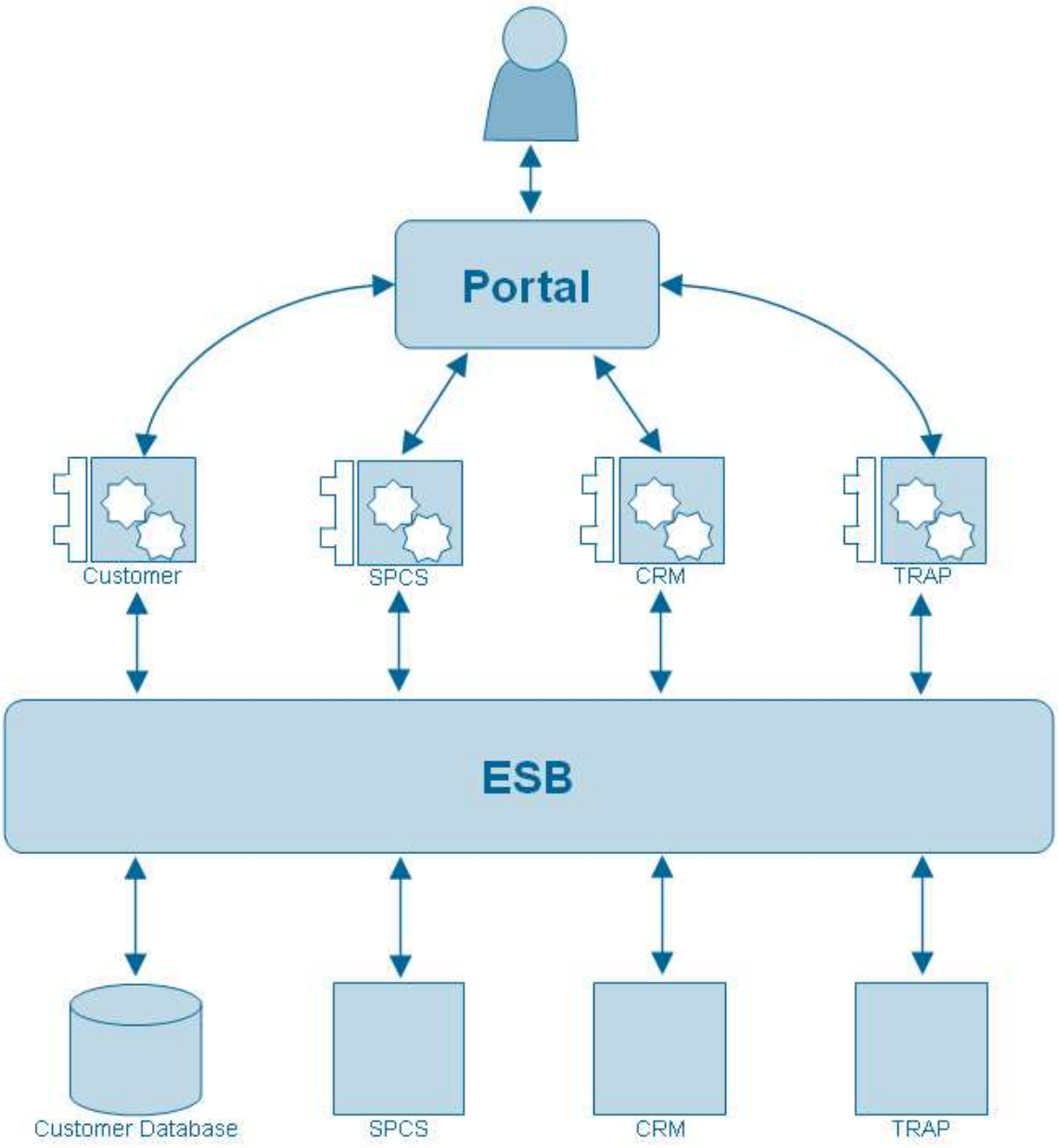

Figure 11. Future Architecture

7.4 First Step – The Prototype

For a thesis on this level, it is a huge task to develop a full SOA system. Therefore, my supervisor and I selected a good start point for this implementation project. We chose to begin implement the master database for customer data and parts of SPCS involving customers, see Figure 12.

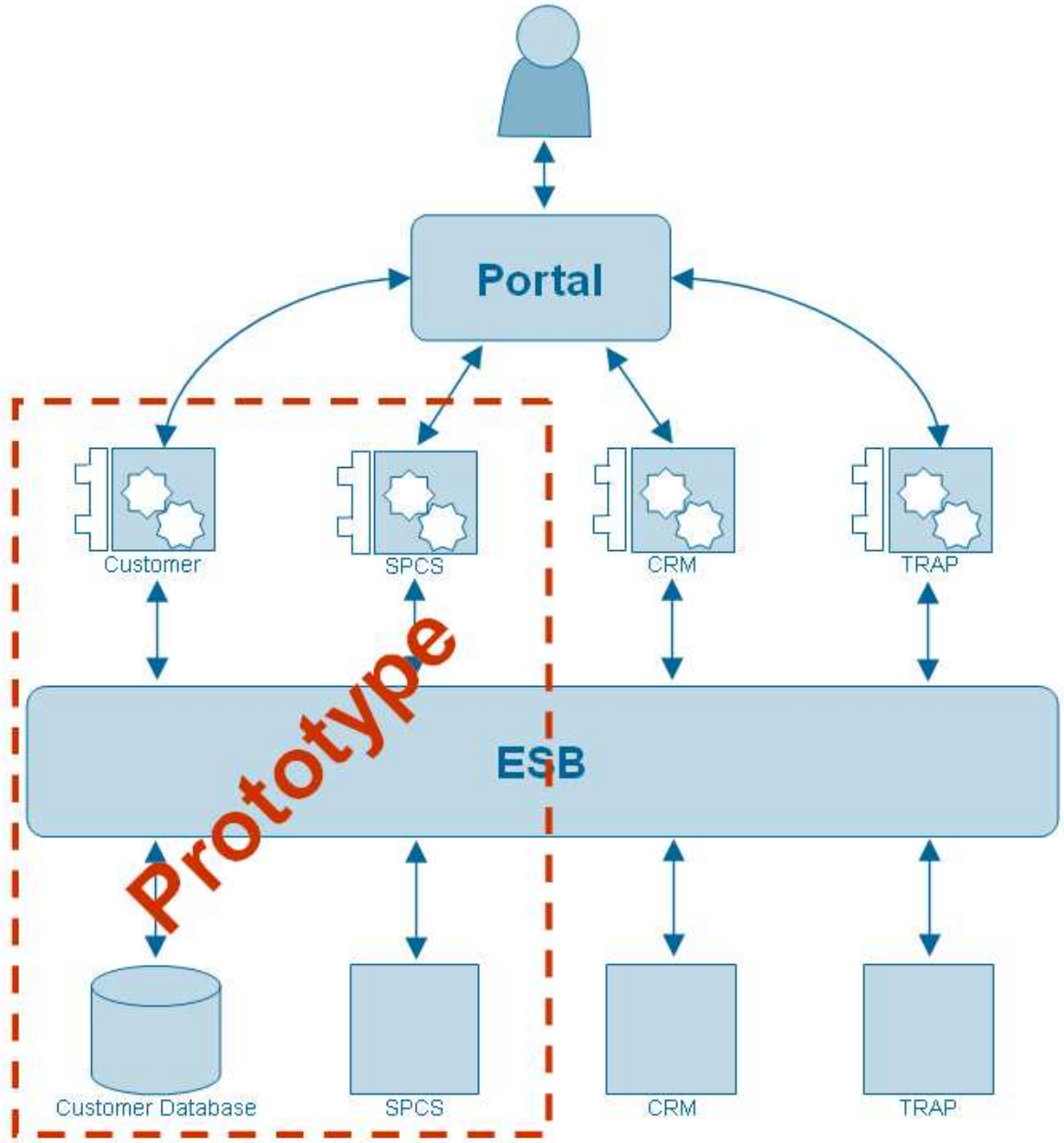

Figure 12. Prototype Architecture

7.5 Implementation

7.5.1 Customer Web Service

The Customer Database Web Service has two functionalities; search customer and add customer. Search customer takes a customer name and returns a list of possible similar customers and its information, while add customer takes some customer information and returns the assigned ID. If the customer already exists, an error code will be returned instead of the customers ID.

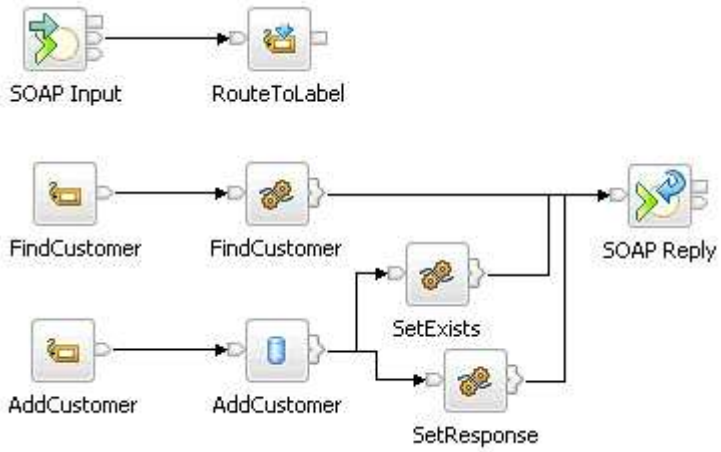

Figure 13. Customer Database Web Service

7.5.2 SPCS Web Service

A Web Service to add customer to SPCS has been developed. The SPCS Web Service have just one function, it is to add a customer to SPCS. This Web Service takes the same input as the add customer function in Customer Database Web Service, thus some customer information. It will return an error code if something went wrong.

This Web Service transforms the message between different formats since the input and output is XML but SPCS is written in C and uses a comma separated format.

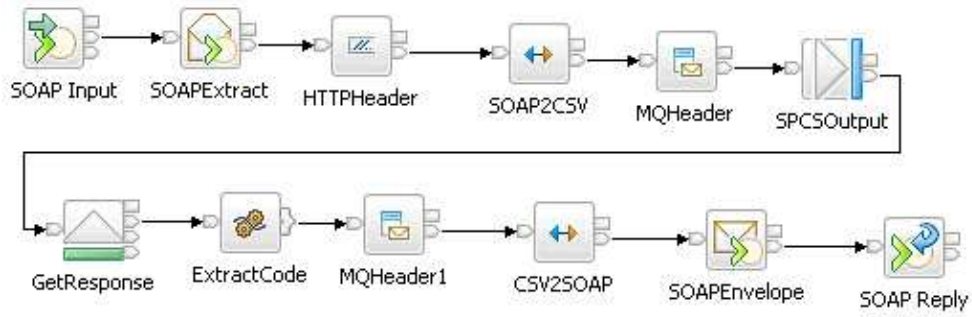

Figure 14. SPCS Web Service

7.5.3 Master Database for Customer Data

In this case, the master database for customer data is represented as a simple regular sql database. Interactions with the master database in this prototype are made through Customer Web Service.

7.5.4 SPCS

To present SPCS as a Web Service some sort of adapter is needed. There are commercially available adapters to buy for SPCS, but in this case my supervisor made an adapter to perform the tasks required. Integration with SPCS in this prototype is made through SPCS Web Service.

7.5.5 External Web Page

An external web page has been developed to show how Web Services can be used when implementing a portal. The page is a JavaServer Pages Technology (JSP) page where a user can invoke the Web Services from within a simple user interface.

Figure 15. External Web Page Interface

In this case when a user tries to add a customer to SPCS, the web page will first call the master database to see if there are any similar customers existing. If the customer already

exists, all information about the customer will be copied from the master database to SPCS, see Figure 16.

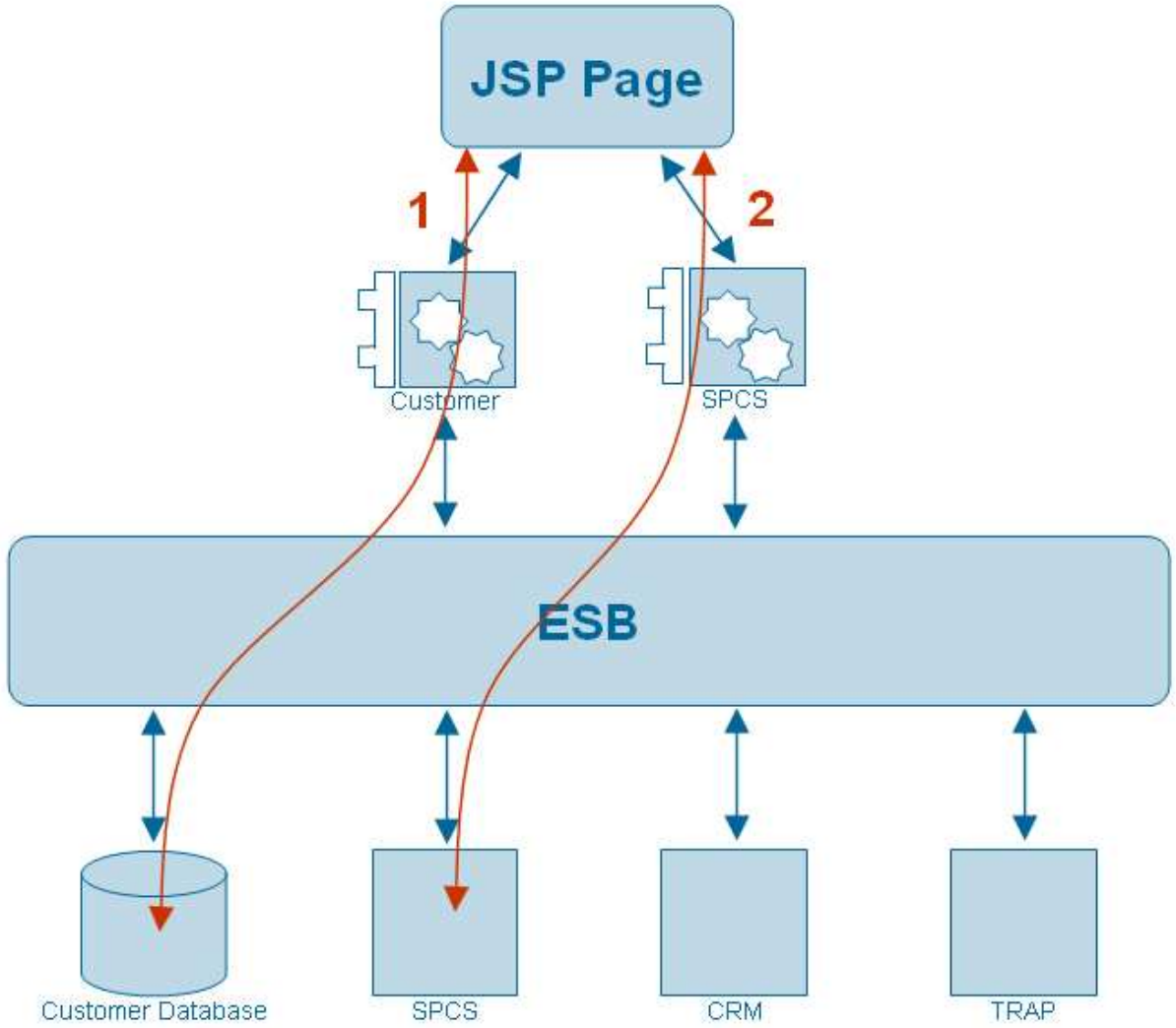

Figure 16. JSP Page Flow

7.6 Results

Although implementation of the prototype is rather small, it shows that SOA facilitates reuse while risks and development time can be minimized by using existing systems, called legacy systems. It also demonstrates that by creating a portal that integrates with legacy systems users are given the impression that there is a single unified IT system to integrate with. Furthermore, it is now possible, if necessary, to quickly add new systems or replace any of the current systems.

Another advantage is that the previously isolated IT systems can now integrate with each other through Web Services or business processes. This opens up new opportunities in the future.

It is important to think before the implementation of an SOA system. If no planning is made before, there is a great risk that the advantage of reusability can not be fully achieved.

7.7 Future development

A potential development can of course be to implement the future architecture shown in Figure 11. After studying SOA, there are other things that could be added to the future architecture to get an even more effective SOA system. For example, a number of business processes and a SOA Registry could be implemented, see Figure 17. This could lead to less user interaction, easier for developers to know what Web Services exists and to facilitate dynamic lookup and use for services.

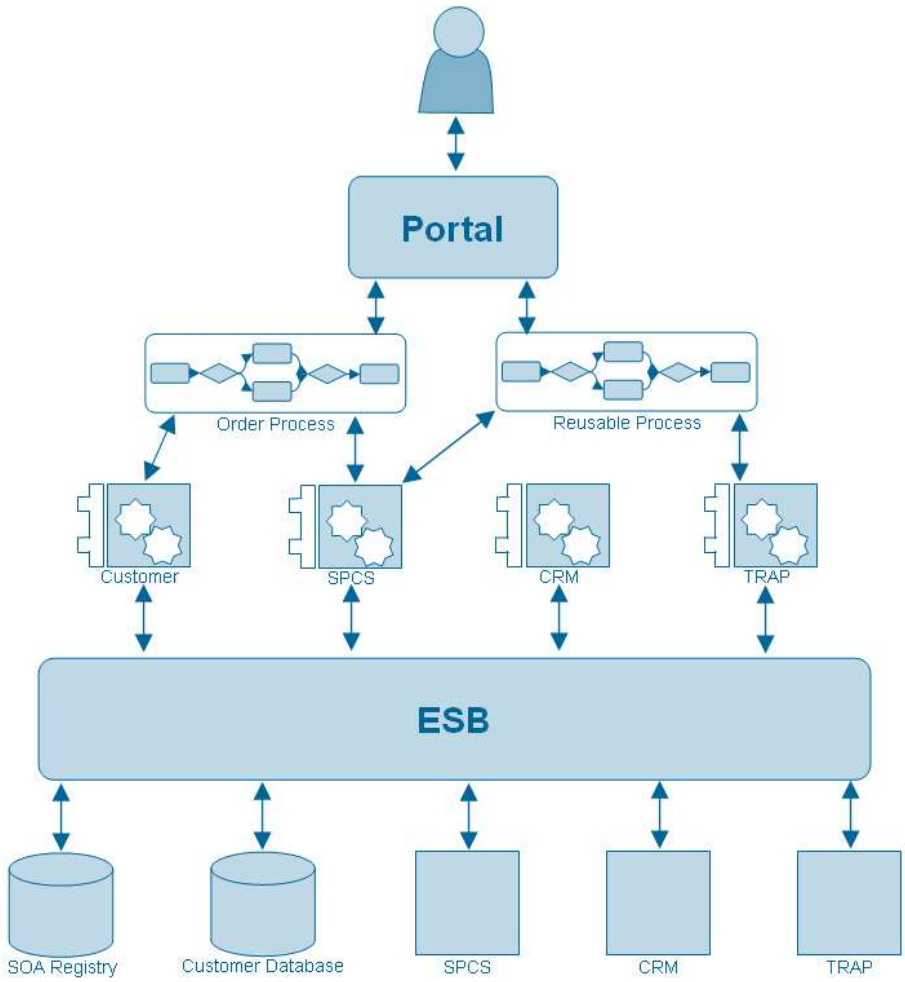

Figure 17. Revised Future Architecture

An example of a useful business process can be an order process. An order process would make it easier for the user and reduce user interaction. The great advantage is that the users do not need to know if the customer already exists in SPCS or not; the order process adds the customer to SPCS if needed. The process could look like in Figure 18.

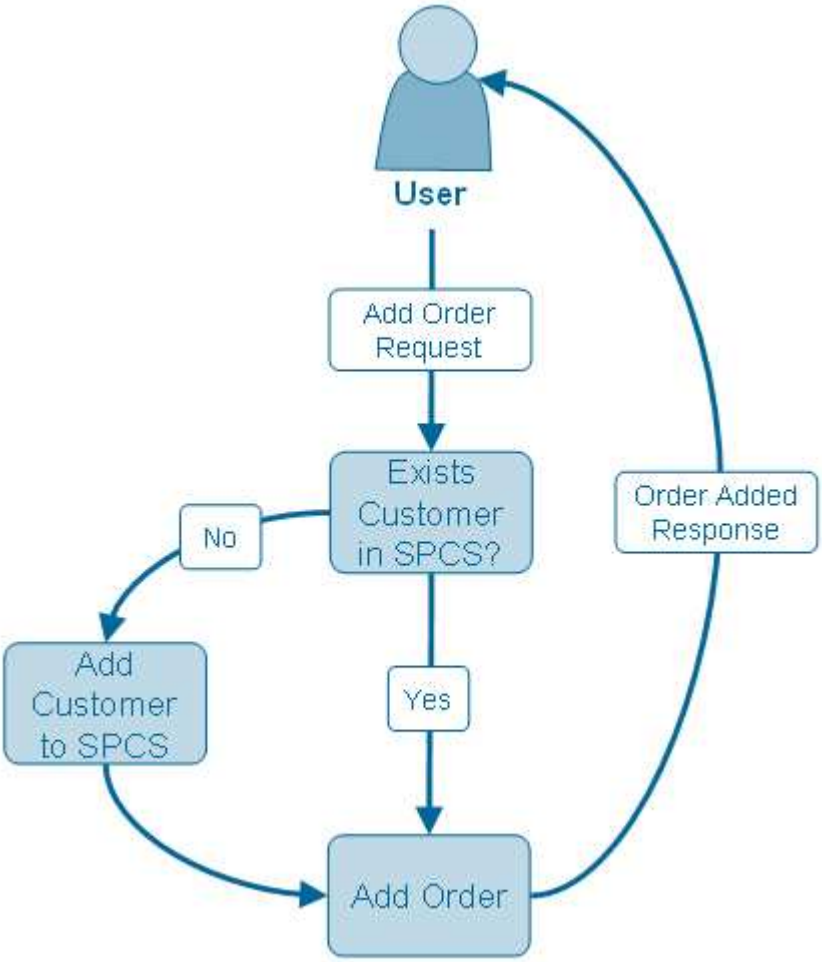

Figure 18. Order Process Sample

8 Conclusion

Is it possible to develop an IT system with the help of SOA where development costs are kept low, the risk are minimized, and where it is possible to see results within a short time of period? It was the main question to investigate when this thesis work started. Since many IT managers believe that the costs, risks and time are major obstacles for IT projects, it was a legitimate question to answer.

Imagine a big car manufacturer. It has a number of different models and has planned to build a new model. Would they build everything from scratch and invent the wheel again, or would they reuse and improve parts that already exist from previous models? You probably choose the second option, reuse and improve.

Regarding IT systems, the first option is the traditional way to develop IT; invent and build everything from scratch for every new IT system. Since the second option is preferable, IT projects should use this approach. SOA facilitates this!

SOA systems are based on reusable services and business processes which means that the same functionality does not need to be in several places. SOA advocates reusing old systems, called legacy systems, by presenting them as well-defined services. The reuse of old, well-tested, systems mean less development and minimized risk.

Reuse leads to less development and minimized risk which in turn leads to shorter development times and faster profit. SOA development reduces overall IT costs in the long term by 20 % compared with traditional methods according to Garner. (Roch, 2006)

All this means that it is possible to develop an SOA system where development costs are lower than traditional methods, that risks are minimized and where it is possible to see results in a short time of period.

The prototype implemented shows that SOA facilitates the reuse of current, well tested systems. Moreover, the prototype shows that already in a rather small systems, multiple applications reuses and share the same Web Services and data which reduces development times and related costs of systems.

But this is not one of the greatest benefits of SOA; still a great side effect. An SOA project should deliver a good value for the business; the major force behind an SOA effort should therefore be to create new and better opportunities for the business.

As the world stands today with rapid change, fierce competition and an ever-flatting world economy increases the importance of flexible enterprises. SOA has the ability to develop and modify services quickly to meet the needs of today. This leads to flexible and thus winning enterprises.

It is important to keep in mind that, building a successful SOA system requires that you have a strong support from senior members and management. This is essential to realize a change in an organization; without support from the top, it is difficult to implement SOA initiatives in a good way.

It is also important to think before. Define services; think about what services that can be critical for the core business and precise limits for each service. Start with a well-defined business process and build from there. Go for a small project but have a vision about how it should be in the future. Always begin with one business process at a time, instead of starting with all processes simultaneously.

Another important thing to keep in mind is that SOA is architecture, not a technology. Like any other architecture you will find value over time; often during the second or third SOA project. It is important to think long term.

9 Index

9.1 Figures

9.2 Examples

9.3 References

Bagwell, Don (2007). *A Practical Introduction to Web Services, SOA and ESB on z/OS*. Available: <http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS2537> (2008-10-24)

Beal, Vangie (2005). Understanding Web Services. Available: <http://www.webopedia.com/DidYouKnow/Computer_Science/2005/web_services.asp> (2008-10-27)

Bedredinov, Rustam & Bedredinov, Timur (2008). *Innovative Systems to Improve Business Processes*. Available: <http://bedredinov.com/> (2008-10-23)

Erl, Thomas (2005). *XML ans SOA*. Available: <http://www.idealliance.org/proceedings/xml04/papers/91/xml_soa_paper.html> (2008-10-27)

Hanson, Jeff (2002). Take advantage of the benefits of loosely coupled Web services. Available: <http://articles.techrepublic.com.com/5100-10878_11-1050425.html> (2008-10-24)

Harding, Chris (2007). *What Makes SOA Different?*. Available: <http://www.ebizq.net/topics/soa/features/8588.html> (2008-10-24)

He, Hao (2003). *What is Service-Oriented Architecture*. Available: <http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html> (2008-10-27)

Hurwitz, Judith, Bloor Robin & Baroudi, Carol (2006). *Service Oriented Architecture For Dummies*. Indianapolis, Indiana : Wiley Publishing Inc.

Höij, Magnus (2008). Tjänstearkitektur mot lågkonjukturen. Available: <http://cio.idg.se/2.1782/1.184047/tjanstearkitektur-mot-lagkonjukturen> (2008-10-27)

IBM (2008). *IBM*. Available: <http://www.ibm.com/> (2008-10-24)

IBM – developerworks (2008). *IBM – developerworks*. Available: <http://www.ibm.com/developerworks/> (2008-10-24)

IBM – Smart SOA Approach (2008). *IBM – Smart SOA Approach*. Available: <http://www-01.ibm.com/software/solutions/soa/smartsoa/ > (2008-10-24)

iSchool (2008). iSchool. Available: <http://ischool.tv/> (2008-10-29)

Microsoft ESB (2008). *Microsoft on the Enterprise Service Bus (ESB)*. Available: <http://msdn.microsoft.com/en-us/library/aa475433.aspx> (2008-10-29)

Microsoft SOA (2008). *SOA & Business Process*. Available: <http://www.microsoft.com/soa> (2008-10-29)

Roch, Eric (2006). *SOA Benefits, Challenges and Risk Mitigation*. Available: <http://it.toolbox.com/blogs/the-soa-blog/soa-benefits-challenges-and-risk-mitigation-8075> (2008-10-24)

Röhne, Jon (2008). 8 kännetecken för vinnande SOA. Available: <http://cio.idg.se/2.1782/1.183677> (2008-10-24)

SearchSOA (2007). SOA Registry. Available: <http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci1270465,00.html> (2008-10-27)

Service Level Agreement (2008). *The SLA Toolkit*. Available: <http://www.service-level-agreement.net/> (2008-10-27)

SLA Zone (2008). *The Service Level Agreement*. Available: <http://www.sla-zone.co.uk/> (2008-10-27)

SOA & WS (2008). *Web Service Standarder*. Available: <http://dsv.su.se/soa/standarder.shtml> (2008-10-27)

Sonic (2008). Enterprise Service Bus (ESB). Available: <http://www.sonicsoftware.com/solutions/service_oriented_architecture/enterprise_service_bus/index.ssp> (2008-10-27)

Sun SOA (2008). *Service-Oriented Architecture (SOA)*. Available: <http://www.sun.com/products/soa/> (2008-10-29)

Thurfjell, Karin (2008). *CIO Sweden: Undvik de sju hindren i IT-projektets väg*. Available: <http://www.idg.se/2.1085/1.159861> (2008-10-24)

W3C – SOAP (2000). *Simple Object Access Protocol (SOAP) 1.1*. Available: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/> (2008-10-27)

W3C – SOAP (2007). *SOAP Version 1.2*. Available: <http://www.w3.org/TR/soap12-part1/> (2008-10-27)

W3C – XML (2001). *XML i 10 punkter*. Available: <http://www.w3c.se/resources/office/translations/XML-in-10-points_sw.html> (2008-10-27)

W3C – XML (2008). Extensible Markup Language (XML). Available: <http://www.w3.org/XML/> (2008-10-27)

Webopedia (2008). *What is Web Services?*. Available: <http://www.webopedia.com/TERM/W/Web_services.html> (2008-10-27)

XML Coverpages (2008). *Universal Description, Discovery, and Integration (UDDI)*. Available: <http://xml.coverpages.org/uddi.html> (2008-10-27)