Master Thesis
Computer Science
Thesis no: MCS-2008-16
Month: Jan Year: 2008

Service Oriented Architecture & Web Services

Guidelines for Migrating from Legacy Systems and Financial Consideration

Adeyinka Oluwaseyi

Department of
Interaction and System Design
School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author:
Oluwaseyi Adeyinka
Address: Snapphanevagen 6A, LGH 077 SE-371 40 Karlskrona Sweden.
E-mail: coashee55@yahoo.com

University advisor:
Jenny Lundberg
Department of Interaction and System Design

Department of
Interaction and System Design
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 102 45

ABSTRACT

The purpose of this study is to present guidelines that can be followed when introducing Service-oriented architecture through the use of Web services. This guideline will be especially useful for organizations migrating from their existing legacy systems where the need also arises to consider the financial implications of such an investment whether it is worthwhile or not. The proposed implementation guide aims at increasing the chances of IT departments in organizations to ensure a successful integration of SOA into their system and secure strong financial commitment from the executive management. Service oriented architecture technology is a new concept, a new way of looking at a system which has emerged in the IT world and can be implemented by several methods of which Web services is one platform. Since it is a developing technology, organizations need to be cautious on how to implement this technology to obtain maximum benefits. Though a well-designed, service-oriented environment can simplify and streamline many aspects of information technology and business, achieving this state is not an easy task.

Traditionally, management finds it very difficult to justify the considerable cost of modernization, let alone shouldering the risk without achieving some benefits in terms of business value. The study identifies some common best practices of implementing SOA and the use of Web services, steps to successfully migrate from legacy systems to componentized or service enabled systems. The study also identified how to present financial return on investment and business benefits to the management in order to secure the necessary funds. This master thesis is based on academic literature study, professional research journals and publications, interview with business organizations currently working on service oriented architecture. I present guidelines that can be of assistance to migrate from legacy systems to service-oriented architecture based on the analysis from comparing information sources mentioned above.

Keywords: Web Services, Service Oriented Architecture, Legacy Systems, Governance.

CONTENTS

1 INTRODUCTION

It is obvious that the world is now a global village enabled by improved information technology and further increases in the society's dependence on computing appear inevitable. Consequently upon this fact, the study of trends and developments in the technology industry cannot be overemphasized. The application of technology to business processes has to a great extent changed the structure and mode of operations in organizations. However the critical path to success is not the technology itself, but its effective application to various elements of the business, as the failure of a new technology is not in the elements of the technology itself but its application. Several technologies which would have benefited organizations by improving their business processes failed as a result of the way those technologies were applied at that point in time. Why are new technologies been developed in the Information Technology circle everyday despite the ample of them that we have around? Over the past decades, the IT industry has been battling with the maintenance cost associated with mainframe managed systems which are a considerable drain on IT department budgets in any business organization. These high costs are results of interactions between the outdated platforms and non-mainstream languages on which these legacy systems are built. Despite this weakness, fewer organizations are ready to take the step towards change while some don't want to even think about it. The main problem is that these systems had always been considered too valuable and costly to risk modernized or changed.

In a typical IT budget survey, the proportion of maintenance costs spent on maintaining legacy systems was very high with figures of up to 90 percent spent on code maintenance during the year 2000 [18]. These costs are increasing on an alarming rate and are a testament to the value of legacy mainframes that they have endured. This implies that few or no developmental projects can be embarked upon if such percentage was used on maintenance purposes only. As the world moves towards the idea of aligning business and IT, it became crystal clear that these systems are not capable of delivering the expected result of business demands. But at the same time, business organizations are keen on getting more out of their earlier investments than developing entirely new systems. For effective result in today's business operations, business or application logic and data need to be separated from each other, a concept which is not obtainable in the existing systems. Therefore the need for a technology that separates business or application logic from data and treats them as services or component based systems. Since organizations still want to benefit from their existing system, an upgrade exercise can be carried out for those systems to meet current business demands. Organizations need assurance that any investment they make will in no time yield some benefits in terms of cash or improved business processes. It is therefore very important that upgrading a legacy system or modernization proposals be prepared and presented in a manner that appeal to the dual needs of improving system quality and delivering business value, and in some cases personal value.

2 THESIS OBJECTIVE

This thesis aims at developing a guideline that can be used when introducing service-oriented architecture through the use of Web services especially when migrating from legacy systems and financial implications for such an investment. Adopting any form of technology, architecture or services in organizations require a serious thought. For example, the requirements that must be fulfilled before adoption in terms of resources, the impact it will have on the organization both now and in the future among others. For the purpose of developing this guideline, the thesis aims at answering the following research questions:

- Despite the potential ability of Service-oriented architecture to align and transform IT and the business world, why is it that the number of organizations that have implemented it is still very low?
- How can a legacy system be migrated to SOA?
- How does the introduction of SOA affect legacy systems?
- Why Web services approach of implementation is considered relative to other available technologies?
- Why making a good financial presentation of SOA investment is considered important to its success?

The above research questions will serve as guide when exploring various aspects of service-oriented architecture and Web services as well as the return on investment. This will be achieved through research into relevant academic literatures, published articles, journals and a couple of interviews with professionals engaged in the use of Web services and SOA. A careful analysis of information will be carried out to compare the information presented from these sources and will further be developed to formulate the guideline that will be the result of this research study. This guide I believe will offer tremendous information and assistance to organizations with the intention of introducing service oriented architecture and help them to avoid pitfalls encountered by early adopters through in depth understanding of some best practices and realize the full benefits of service oriented architecture. Organizations that have already introduced SOA can also benefit from this guide for further improvement and maintenance. There is no technology available today that thus not have its own weaknesses or areas that need to be improved upon. The results presented in this research is not sufficient to cover all the areas related to implementing service-oriented architecture using Web services, therefore further areas of study will be identified especially the weak areas where detailed study can be carried out for improvement.

3 RESEARCH METHODOLOGY

To carry out the research work, a number of research questions relative to the research area were designed with the aim of providing corresponding answers. This will be achieved through qualitative method of research. The first step taken was to have an in-depth knowledge into the research topic through the study of academic literatures, scientific articles and journals, and industry professional publications. This was followed by interviews sections with professionals that were currently working on different aspects of service-oriented architecture and Web services. A constructive analysis was then carried out by comparing the information obtained from literature study and practical information obtained from experts in the industry. The guideline presented in this report is based on the critical view of the comparison and analysis. However, a limitation to this study from the practical point of view is the limited access to companies within Sweden which is a result of the language barrier.

Different academic literatures and journals established the fact that implementing SOA does not necessarily solve all problems out there, but has a lot of advantages and benefits compared to existing systems. The most emphasized benefit is the ability to position organizations in a way to respond to future business requirements and challenges because of its flexibility. I discovered that there are no clear industry leaders as it were but IBM and Microsoft have been contributing in most cases to establishing standards together with standard bodies formed like WS-I. IBM has implemented a SOA to help boost its business capacity which has been a source of inspiration to many other organizations to follow suit.

Information gathered during the interview process revealed that most organizations are yet to adopt the concept of SOA for many reasons. The most obvious ones being the cost and risk associated to migrating from their legacy system because these systems are considered to be the pivot generating profit for the organizations. Secondly, the acclaimed benefits of SOA are not so evident from the perspective of those that have implemented it coupled with some unsuccessful implementations. Also, there are not enough skills and expertise yet developed in this area which makes organizations to rely on external professionals or consultants. From the professional's perspective, it is not every organization that is in need of SOA. Organizations must ensure their business capability really fits in before embarking upon such huge project. Also, though there are several ways of carrying out a SOA, the wide acceptance of Web Services was majorly based on the evolution of associated industry standards like XML, WSDL, SOAP and UDDI and the World Wide Web.

4 BACKGROUND

Several industry trends are converging to drive elemental IT changes around the concepts and implementation of service orientation [10]. These key technologies are; Extensible Markup Language (XML), a common independent data format across enterprise, Web services, an XML-based technology for messaging, service description, discovery and extended features. Others are business process management which is a methodology and technology for automating business operations, and Service-oriented architecture a methodology for achieving application interoperability and reuse of IT assets.
Service-oriented architecture is built upon a tradition of technology and a progression of business needs. It focuses on reusable code and modular design, objects, components, and enterprise application integration [31]. SOA is an emergence of increased data and application integration, strategic agility and flexibility in the business sector. Seen as the next innovation within the IT market place, vendors and business organizations are anticipating the potentials and its enormous impact. According to a survey carried out by cutter consortium on SOA adoption and best practices in organizations, 64% of the respondents were either in the process of deploying or are thinking about deploying an SOA while 10% had already deployed it. To establish its importance within the corporate IT environment, few examples of organizations that have benefited from the deployment of SOA are given. McGraw-Hill Education, in an effort to deliver more relevant content through online textbooks, saw an increase in revenue following the deployment of service architecture. Also, Sabres Holdings by managing services more effectively, reduced the cost necessary to deliver access to new and existing customers. In the same vein, Sprint during the implementation of a service repository, gained new business that can directly be attributed to SOA deployment [22]. International Business Machine (IBM) also obtained business transformation enabled by service-oriented architecture.

4.1 WHAT IS SERVICE ORIENTED ARCHITECTURE?

Service-oriented architecture is about the evolution of business processes, applications and services from today's legacy-ridden and smooth integration of disparate applications to a world of connected businesses, accommodating rapid response to change and utilizing vast degrees of business automation. It is a set of general design principles that enables organizations to change business processes on the wing and respond to the shifting demands of the business in a manner that would be impractical or cost-prohibitive using conventional application development and resources allocation [10]. SOA can be viewed as a computing methodology or approach to building IT systems in which business services i.e. services provided by an organization to clients are the key organizing principles used to align IT systems with the needs of the business. Earlier approaches used in building IT systems focused on direct use of specific implementation environments such as object orientation or procedure orientation to solve business problems. These approaches resulted in systems that are often tied to the features and functions of a particular execution environment technology. From the above description of service-oriented architecture, it shows clearly that service is a key component. A service can be considered as a means by which the needs of a consumer are brought together with the capabilities of the service provider [22]. Services within an organizational context can either be driven by the needs of the consumer, user/business requirements and drilling down to the system level (top down), or taking into account the system capabilities of the service provider and building up services that can be exposed to the higher layers in the architecture (bottom up). But as it is today, services are more built from the engineers or suppliers view than that of the users.

The interest in SOA as a guiding principle was as a result of the IT community shifting away from large scale development of applications and towards the creation of services that more accurately reflect underlying business processes [22]. The business and IT sector now complement and needs each other than ever before. But over the years, the successful integration of these two sectors has been a nightmare even with the emergence of different technologies. While previous technologies have not efficiently enabled the IT/business-unit relationship to improve, it is the belief of researchers and IT professionals that the nature of services as a consumable product represents what could be the much needed shift. The major difference between service-oriented development and previous approaches is that service orientation focuses on the description of the business problem, while previous approaches focus more on the use of a specific execution environment technology. The technique with which services are developed enhanced their alignment to solving business problems than was the case with previous generations of technology.

4.2 BENEFITS OF SOA

The major reason for the emergence of SOA is for the relationship between IT and the business units to improve. Business organizations are dealing with two fundamental concerns; the ability to quickly change to meet today's urgent demand for new level of agility, and the need to reduce cost [22]. To remain competitive, businesses must adapt quickly to internal factors such as acquisition and restructuring, or external factors like competitive forces, customer requirements or government regulations. Cost-effective, flexible IT infrastructure is highly needed to support the business. The concept of service oriented architecture can help organizations succeed in the dynamic business landscape of today. This can be achieved through the primary characteristic of SOA which encourages the reuse of business logic. SOA, when properly implemented, makes reusability extremely cost-effective. The motivations for different service oriented architecture initiatives include a range of technical and business reasons. The most common motivations are agility, flexibility, reusability, data rationalization, integration, and reduced costs.

- **Leverage Existing Assets**

SOA provides a layer of abstraction that enables an organization to continue leveraging its investment in IT by wrapping these existing assets as services that provides business functions. Organizations potentially can continue getting value out of existing resources instead of having to rebuild a new system from the scratch if they could employ an effective migration path from the legacy systems to a service based system.

- **Easy Integration and Complexity Management**

The integration point in Service Oriented Architecture is the service specification and not the implementation. This provides implementation transparency and minimizes the impact when infrastructure and implementation changes occur [22]. By providing a service specification in front of existing resources and assets built on disparate systems, integration becomes more manageable since complexities are isolated. This becomes even more important as more businesses work together to provide the value chain.

- **Responsively and Faster Time to Market**

The ability to compose new services out of existing ones provides a distinct advantage to an organization that has to be agile to respond to demanding business needs. Leveraging existing components and services reduces the time needed to go through the software development life cycle of gathering requirements, performing design, development and

testing. This leads to rapid development of business services and allows an organization to respond quickly to changes and reduce its time-to-market.

- **Reduce Cost and Increased Reuse**

With core business services exposed in a loosely coupled manner, they can be more easily used and combined based on business needs. This means less duplication of resources, more potential for reuse, and lower costs. Reuse seems to have been the holy grail of software for decades [22]. With effective service-based software reuse programs in place, IT delivery organizations can build up libraries of business meaningful functionality that are not attached to particular usage settings, and are easily composable and re-composable to meet new business requirements which can be hosted remotely or locally. These libraries can help organizations to reduce the investment required to address new business software requirements, make delivery of new solutions faster and more dependable. It will also improve the accuracy and speed with which solutions can be altered.

With the numerous benefits of object orientation and components such as provision of a better paradigm for development of complex software systems, distribution, scalability and redundancy, the reuse problem was not adequately solved. Today, the use of services brings back the hope for solution. Services provide a larger-granularity runtime unit of functionality and reuse. The most important value of reuse is consistency. SOA allows separate access to functions or data such that every application that needs to make use of the function or data can use the same service to get it. Most enterprises suffer from redundant data or applications. Imagine an enterprise-wide customer service that would manage the shared customer information (such as an address) for all systems so that the information would need to be changed only once. SOA provides an approach for consistency of processes and data for both internal and external customers.

- **Improved Flexibility**

Flexibility concerns the ability of solutions to be altered in the face of changing business and technology requirements. This is boosted by the loosely-coupled nature of services which are composed to meet solution requirements in a SOA environment. Flexibility is a key to the change management element of IT-business alignment. Since the business environment is highly dynamic and volatile, SOAs allows businesses to be ready for future challenges. Business processes which comprise of a series of business services, can be more easily created, changed and managed to meet the needs of the time. SOA provides the flexibility and responsiveness that is critical to businesses to survive and thrive.

- **Division of Responsibility**

Service-oriented development aims to separate business logic from the data thereby gives the ability to more easily allow business people to concentrate on business issues, technical people to concentrate on technical issues, and for both to collaborate using the service contract.

The existing mainframe systems also called legacy systems are not capable of some of the mentioned benefits. For example, the cost of maintenance a legacy system is very expensive and does not support critical business application because business logic and data are not separated.

4.3 CHALLENGES OF SOA

I will like to stress this point that an SOA is not a silver-bullet solution for all problems when it comes to IT and business integration. Implementing an SOA is good but should not be expected to suit every IT and business domains, sometimes a different kind of architecture is more suited to solve some problems. For example if you have hard real-time or near-real-time requirements, an SOA approach always introduces certain latency [10]. The technological risk of SOA can be challenging due to factors like early adoption and evolution of supporting technology, distributed infrastructure which requires high availability and scalability, organizational change since SOA crosses system boundaries, efficiency in development and reuse, entity aggregation e.t.c. Also, new competences must be developed spanning project management, development and operations, analysis and design. Successfully solving problems and providing useful enterprise applications requires a combination of business, technology, architecture, organization, people and process. Relative to successful implementation of SOA, technology is the least important factor to be considered. Some of these factors are expatiated below:

- **Efficiency in Development and Reuse**

Making development more efficient depends on a variety of factors, such as the reuse of services and the ability to quickly compose applications from those services which in turn requires a different approach to service and application development. Service developers must create services that fit into the overall architecture and conform to the enterprise business and information models [22]. When there is need for enhancement of services, it has to be done in a controlled fashion that maintains the integrity of the service architecture and design. This in turn must conform to versioning and compatibility requirements which make the job of application developers easier. Furthermore, methods and tools for modeling and composing business processes from existing services need to be established with organizational changes to support service development and use across the enterprise. Addressing the concept of reuse as a way to reduce development has some costs implications and time to market.

- **Integration of Applications and Data**

The integration of existing applications and data is perhaps the most perplexing problems and one of the top priorities of IT organizations for over a decade. Earlier solution to resolve application integration through Enterprise Application Integration (EAI) yielded little result because, too often, fragile and in-maintainable solutions have been put in place that created point-to-point connections over a variety of different technologies and protocols [22]. Instead of connecting individual applications directly together, providing services that connect individual applications into the overall enterprise makes integration of applications easier to manage. Also, data integration is extremely difficult and various attempt to implement a global enterprise data model failed. For services to fit together into a business process or to be composed together in a meaningful way, they have to share a common data model and semantics [22]. They do not have to agree on every single item and field of data, they only have to agree on what the shared enterprise-wide data should be.

- **Agility, Flexibility, and Alignment**

Agility and flexibility occur when new processes can quickly and efficiently be created from the existing set of services. Achieving agility and flexibility requires an easily searchable catalog that lists the functions and data provided by the services. These services must share and conform to a common enterprise semantic model. Alignment is “ensuring that the services fit within a broader IT framework, both in the relationship to the business processes

and the strategic plan" [22]. To achieve alignment, SOA can adopt an enterprise architectural approach which needs a business architecture that lays out a roadmap for the processes and services of the enterprise now and over time, and that identifies the functional and application capabilities to support the services. It needs an information, application and technology architecture that define what the technologies are and how they are used to support processes, services, integration, data access and transformation, and so on. There has to be a process that directly integrates the enterprise architecture (business, information, application, and technology) into the development process. Of utmost importance is an organizational and governance structure to support and enforce these processes.

4.4 Bests Practices of SOA

The Project Management Institute defined best practice as a technique or methodology that through experience and research has proven to reliably lead to a desired result. A commitment to using the best practices in any field is a commitment to using all the knowledge and technology at one's disposal to ensure success. Best practices advocate that successful SOA implementations most often take place within the context of an organizational commitment to operate more efficiently and effectively [12]. Best practices of reference architecture, common semantics, governance, business process modeling, design time repository, and model based development are aimed at enabling agility, flexibility, and alignment which supports building SOAs that meet the goals and expectations of today's enterprises.

- **Reference Architecture**

Having and maintaining reference architecture is one of the more important but difficult best practices for SOA which is also an important critical success factor in achieving SOA goals. The reference architecture represents a more formal architectural definition, one that can be used for objective validation of services and applications [22]. It defines the layers, architectural and design decisions, patterns, options and architectural building blocks of SOA that is services, components, and flows that collectively support business processes and goals. A typical SOA reference architecture should incorporate the following:

- Support for enterprise architecture concepts, particularly the sub architectures of business, information, application, and technology.
- Specification of a hierarchy and taxonomy of services and service types.
- Separation between business, application, and technology concepts.
- Integration into the development process.

Using reference architecture as a guideline, organizations can develop a road map for service-oriented architecture security and management meeting current enterprise needs and supporting future large-scale deployments of services.

- **Common Semantics**

Defining a common enterprise semantic and information model is vital to achieving agility and flexibility. These can be achieved by consolidating redundant systems, simplifying integration using semantic exchange models, examining standards and adopting tools that utilize metadata to support the entire integration life cycle. Without common semantics, services cannot be easily combined to form meaningful business processes. The common semantics should be able to identify information that must be shared across the enterprise and between services. It must also define the meaning and contexts of that information and identify techniques for mapping enterprise semantics to existing application data models.

- **Governance**

Implementing a solution requires the definition of enterprise policies and establishment of strong auditing and conformance mechanisms to ensure that enterprise policies are being adhered to. Governance enforces compliance with the architecture and common semantics, and it facilitates managing the enterprise-wide development and evolution of services [22]. Governance consists of a set of policies that services and applications must conform to, a set of practices for implementing those policies, and a set of processes for ensuring that the policies are implemented correctly. An organizational structure should be in place to define and implement governance. Governance of SOA should achieve the following:

- Include policies regulating service definition and enhancement, including ownership, roles, criteria, review guidelines e.t.c.
- Specify identification of roles, responsibilities, and owners.
- Enforce policies that are integrated directly into the service repository where appropriate.
- Include guidelines, templates, checklist, and examples that make it easy to conform to governance requirements.
- Involve a review of service interface definitions for new services enhancements to existing services. It ensures that the service definition conforms to standards and aligns with the business and info0rmation models. This is typically done by a service review board or the unit responsible for the service.
- Include an architectural review of applications and services to ensure that they conform to the SOA and enterprise architecture, typically done by the architecture review board.

Governance might also support processes that allow different organizations to make changes to shared services. It should not be seen primarily as a review activity but must follow a carrot-and –stick approach with an emphasis on enabling developers to build conforming applications and automating governance activities and policies [22]. The failure to manage the evolving SOA can result in financial loss in costly service redesigns, maintenance, and project delays. More damaging are the potential loss of revenue and the business liabilities.

- **Business Process Modeling and Management**

IT organizations should be convinced by now that without a clear focus on business process, service oriented architecture will not be useful without a business process management infrastructure. Given that a true SOA map needs that business services be created which are independent of each other, it is essential that there be a mechanism in place to enable these components to be linked together. Business processes need to change relatively frequently, yet be based on a stable underlying capabilities [22]. The flexibility to do this comes from being able to quickly construct business processes from business services which are relatively stable. Business processes should have the following:

- Be specified using business process models and executed in a business process management system.
- Be composed of activities that are implemented by business services provided by the SOA.
- Pass information into, out of, and within the process in the form of documents, which are built on top of the common information model.

Service-oriented architecture aims to promote business agility, but that agility depends as much on supporting new efficiencies for people as it does on liberating access to systems and services. Effective business process modeling and management will enable businesses attain

a competitive edge through repeatable and predictable process and compliance execution involving people and systems.

- **Design Time Repository**

Over the time, a common mistake is to confuse a runtime registry with a design time repository. A registry is used at runtime to identify a service endpoint for a requested service interface, while a repository is used at design time to find existing services for inclusion in a process during the design of that process [22]. A design time repository should contain a catalog of available services, provide sophisticated search capabilities for identifying potential services and include metrics on service usage. It should enable capabilities for examining a service, its interface and implementation, its design, and its testing to determine if it is appropriate for the desired usage. Furthermore, the design time repository should provide notification to interested parties of upgrades to service or other events and automate the implementation of certain governance policies.

- **Model Based Development**

Model-based development is a best practice in software engineering in general and is applicable in SOA development. Parallel to developing large systems consisting of multiple interacting subsystems and components, service oriented architectures include the connection of new and existing service components into a specific application. This application must be iteratively developed, analyzed, and tested from a high level down through final implementation and support [22]. This system must account for all parties in the architecture including the intended users and the standard services being tapped by the end application. A model-based development approach for SOA should include the following:

- Provide a higher level of abstraction for software development and the ability to visualize software and service designs
- Support a domain specific-language for the implementation of SOA
- Automatically integrate the SOA reference architecture into the design environment
- Separate the concerns of business, services, and technology

The systems-based approach of model driven development makes development of the entire SOA and its components simpler while ensuring higher initial accuracy and consistency.

4.5 SOA LIFE CYCLE

Service-oriented architecture development is a gradual process that has many processing stages that can be referred to as life cycles. These gateways serve as a compass to follow to achieve an efficient and robust system. The SOA lifecycle is a representation that aims to demonstrate the association and dependencies between various independent processes that is made up of a mature, enterprise SOA program [33]. This includes the conceptualization and initiation stage, planning, development, deployment and continuous support even after implementation. The following constitute the basic stages in the life cycle of a typical service-oriented architecture development.

- **Development Stage**

Mostly, organizations design and build services that match up to precise steps within a business process during the development phase. These services can be combined to produce a composite service or application for implementing specific business functions. The choice

of which service interface to use to make available the services to the organization is then made for example; it can be through the use of Web services interfaces or some others. Since requirements are expected to change overtime, also at this stage preparation for further development after the initial service is deployed is made and the process of managing the changes and cost implications are projected. A good service metadata management and service versioning enables organizations to enhance services and manage the deployment of multiple service versions in a cost effective and productive manner.

- **Integration Stage**

After the service has been designed and the interface has also been developed, the next step is to integrate it with other services or IT systems such as databases, applications and transactional management systems since it is not going to be working in isolation. These integrations mostly demands transformation of data to map between diverse data schemas, as well as dynamic routing for linking the appropriate services at run-time.

- **Orchestration Stage**

After a couple of services have been developed, they can then be combined together step by step to create seamless, reliable process flows. The process of "gluing" services together with flow logic is called orchestration [33].

- **Securing Stage**

It is very important that accesses to services are secured before they are deployed in any form. For example the processes for authorizing and authenticating users, as well as provisioning them and managing their identities, must be designed before sensitive information is exposed as a Web service.

- **Management Stage**

The management stage describes the definition and enforcement of service level agreements for services, and various operational policies like auditing and billing for service usage. Good management policies can guarantee an organization that their services will be most likely reliable, available and constantly monitored for exceptions or failures.

- **Accessing Stage**

Services can be accessed in different ways and are typically exposed to users through a portal or a composite Web application. It can also be accessed through wireless devices such as cell phones and handheld devices. An SOA environment supports multi-channel access to services which enables organizations to adapt user interfaces without modifying the underlying services. This provides the user an increased flexible to access those services.

- **Analysis Stage**

For operational administrators and workers to effectively monitor, analyze and respond to time-sensitive issues, the analysis of services, events and business processes involved in business operations often needs to occur in real time. This enables organizations to figure out difficulties encountered in their processes and inform the concerned personnel when a particular event warrants attention. Following a pattern as such is an approach that ensures proper security, reliability and availability of services.

- **Service-oriented Life Cycle Phases [33]**

4.6 SOA GOVERNANCE POLICY

Service-oriented architecture governance is a major component among the best practices that describes how organization and process tie together the other components of the success equation e.g. architecture, business, and technology. Though the focus of services is not on core administrative functioning but rather on the development of a service that will be consumed by those both internal and external to the firm, without an effective SOA governance policy, enterprises will struggle to achieve the results that they desire. Governance processes must be adaptive and flexible in what the enterprise needs, what systems the enterprise will build, and how those systems will be built will be much different in the future compared to now [2]. Organizations must have discipline and rigor in the enforcement of the architectures, standards, and policies they adopt for SOA. Effective SOA governance should achieve four main goals:

- The deployed services are aligned with the business
- The services enable the business to achieve the benefits desired
- The services are delivered effectively
- The services are owned and orchestrated across the enterprise

Focusing on the above goals, a firm can deploy a governance framework that includes; service ownership, service orchestration, service alignment, service delivery, and service value [22]. These SOA governance areas are the issues that IT executives should focus on as they seek to impact the business with this flexible, agile technology and concentrate on the convergence of three aspects: people, processes, and technology. People are those who have the decision rights to make the necessary types of decisions, processes are how the decisions are made and what mechanisms are used to determine if the goals were achieved. Technology is the facilitating mechanism used to facilitate the people and the processes within these SOA governance elements to make the decisions.

- **Service Ownership**

Service ownership identifies the issue of who is responsible for the development, implementation, maintenance, and enhancement to the service. It is recommended that ownership be clearly delineated and shared between a business owner and an IT owner, and that the nature of the ownership should be negotiated and formalized to ensure service success.

- **Service Orchestration**

Service orchestration refers to an enterprise-wide governance arrangement for examining proposed services to ensure that the needs of the enterprise are effectively being achieved. It prevents duplication of efforts and promotes the reuse of components in an efficient manner. Orchestration can be achieved by cross-functional review boards, an executive advisory board, or a mechanism that focuses on the enterprise rather than the individual business units.

- **Service Alignment**

Service alignment focuses on ensuring simultaneous linkage between the services, business processes and the strategic IT plan. The services must fit within the broader IT framework by the organization to facilitate more effective business processes which require a governance mechanism to ensure that this occurs efficiently.

- **Service Delivery**

Service delivery refers to the governance arrangements designed to facilitate the distribution of services, including the base underlying architecture and infrastructure to ensure success. The execution of the services requires a governance arrangement that ensures reliability and consistency, and also requires that there is a high degree of commitment to delivery success. The delivery is ensured through service-level agreements (SLAs) that are co developed by the business and IT.

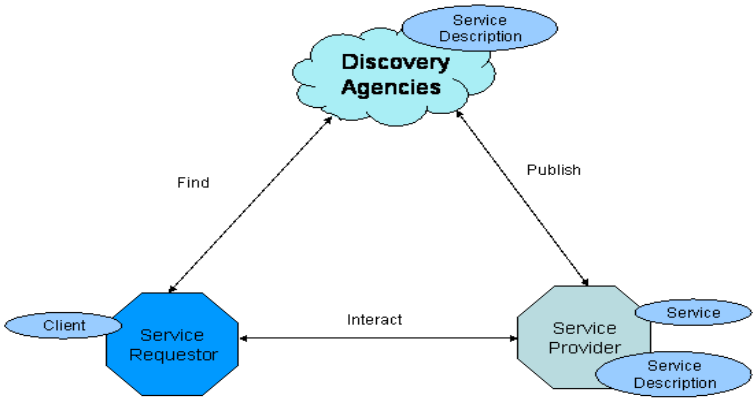
- **Service Value**

Service value is about executing value propositions throughout the service lifecycle, ensuring that IT delivers the value that was originally proposed. This can be accomplished through the use of dashboards and metrics that should be negotiated, and focused on delivering value not only from the cost perspective but from a variety of other perspectives, captured by a balance scorecard.

5 SERVICES AND WEB SERVICES

Service-oriented architecture can be implemented using various technologies like Web Services, Service Component Architecture (SCA), Enterprise JavaBeans (EJB), CORBA and so on. It is possible for a service developed to have different kinds of interfaces. For example, a service can have a web service interface and a Java-based SCA-service interface. However, Web services is the most common new technology for implementing Service oriented Architecture. Despite some current limitations, an SOA with Web services is an ideal combination of architecture and technology for consistently delivering robust, reusable services that support present business needs and that can without difficulty be adapted to satisfy changing business requirements [10]. SOA based on web services aims at simplifying integration by providing universal connectivity to existing systems and data. The W3C's Web Services Architecture Working Group jointly agreed on the following working definition of a Web Service. "A Web Service is a software application identified by a URI, whose interfaces and bindings are capable of been defined, described, and discovered as XML artifacts. A Web Service supports direct interaction with other software agents using XML-based messages exchanged via internet-based protocols [23]. Basic web services combine the power of two ubiquitous technologies: XML, the universal data description language; and the HTTP transport protocol widely supported by browser and web servers.

Web Services = XML + transport protocol (such as HTTP)



Web Service-oriented architecture [30]

5.1 Understanding services

Though services and Web services are commonly used interchangeably in many situations, there exists basic distinction between the two. A service is the observable set of behaviors of a system accessible via a prescribed interface. "A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised in consistent with constraints and policies as specified by the service description" [5]. A Web service is a specific type of service, describing the interface using the WSDL, using SOAP over HTTP as a transport protocol for example. A service is provided by an entity called the service provider for use by others and can be accessed by means of a service

interface where the interface comprises the specifications of how to access the underlying capabilities. Services, much like components, are intended to be independent building blocks that collectively represent an application environment. But different to traditional components in the sense that, services have a number of unique characteristics that allow them to participate as part of a service-oriented architecture. One of these distinguishing features is complete autonomy from other services which means that each service is responsible for its own domain. This design approach results in the creation of isolated units of business functionality loosely bound together by a common compliance to a standard communications framework. As a result of the independence of services in this framework, the programming logic they encapsulate does not need to comply with any particular platform or technology set. The most widely accepted and successful type of service is the XML Web service which has two fundamental requirements. It communicates via Internet protocols (most commonly HTTP) and it sends and receives data formatted as XML documents. Broad acceptance of this design model however has resulted in the emergence of a set of supplementary technologies that have become de facto standards [30]. Therefore, an industry standard Web service is generally expected to provide a service description that at minimum consists of a Web Service Definition language (WSDL) document and be capable of transporting XML documents using Simple Object Access Protocol (SOAP) over HTTP.

There are three basic kinds of knowledge associated with a service; service profiles, service models, and service groundings [5]. A service profile is a description of the offerings and requirements of a service that is, its specification. This specification is essential for a service to be discovered by a service-seeking agent and can help the agent to determine whether a service is appropriate for its purposes based on the service profile. A service model describes how a service works. Such information is important for a service-seeking agent for composing services to perform a complex task, and for monitoring the execution of the service. While a service grounding specifies details of how an agent can access a service. Typically, grounding will specify a communication protocol and port numbers to be used in contacting the service.

The difference between Web services designed for service oriented architecture and Web services created for use with other distributed application environments is that they typically follow a set of distinct conventions. The W3C framework for web services consists of a foundation built on top of three core XML specifications [5]. Web Services Description Language (WSDL) which is a descriptive interface and protocol binding language, Simple Object Access Protocol (SOAP) which is an XML-based remote procedure call and messaging protocol, and Universal Description Discovery and Integration (UDDI) which is a registry mechanism that can be used to look up web service descriptions.

5.2 Key Features of Web services

Some of the features of Web services that have made it to be an industry wide choice are discussed below [3].

- **Web services are self-contained**

For an organization to adopt web services, a programming language with XML and HTTP client support is enough to get such an organization started with no additional software required on the client's side. On the server side, merely a web server and the servlet engine are required. It is possible to use web service to enable an existing application without writing a single line of code.

- **Web services are self-describing**

Neither the client nor the server knows or cares about anything besides the format and content of request and response messages (loosely coupled application integration). The definition of the message format travels with the message with no external metadata repositories or code generation tools required. The only part of the service that is visible to the outside world is what is exposed through the service's description. Outside of what is expressed in this description, the nature or form of the underlying logic is invisible and irrelevant to other services.

- **Web services are modular**

Web services is a technology for deploying and providing access to business functions over the web, while J2EE, CORBA, and other standards are technologies for implementing this Web services. Web services can also be published, located, and invoked across the web.

- **Web services are language independent and interoperable**

The interaction between a service provider and a service requester is designed completely to be platform and language independent. This interaction requires a WSDL document to define the interface and describe the service, along with a network protocol (usually HTTP) [23]. Because the service provider and the service requester have no idea of what platforms and languages the other is using, interoperability is achieved.

- **Web services are inherently open and standards based**

XML and HTTP are the technical foundation for web services. A large part of the Web service technology has been built using open source projects. Therefore, vendor independence and interoperability are realistic goals.

- **Web services are dynamic**

Dynamic e-business can become a reality using Web services because, with UDDI and WSDL, the web service description and discovery can be automated.

- **Web services are composable**

Simple Web services can be aggregated to complex ones, either using work flow techniques, or by calling lower-layer Web services from a Web service implementation. This allows logic to be represented at different levels of granularity and promotes reusability and the creation of abstraction layers.

5.3 Web Services Interoperability and Organization

Web services are one of the rising stars in the IT world, supporting the integration of existing systems and sharing of resources and data, both within and outside an organization. Web services specifications progress toward standardization through a variety of ways, including small groups of vendors and formally chartered technical committees. For the key promise of Web services interoperability to work, standards need to be carefully managed. Also guidance in interpretation and implementation of standards is essential to facilitate adoption of a technology. The Web services Interoperability Organization has an important role as standards integrator to help Web services advance in a structured and coherent manner. Such

an organization that is committed and has actively participated in the WS-I standards development and early delivery of WS-I compliance in runtime and development products is IBM. Microsoft and IBM are the de facto leaders of the Web services specification movement and have defined or assisted to define all the major specifications [28]. WS-I standards and guidelines are seen as an enabler for Web service interoperability. Web services Interoperability Organization (WS-I) is an open industry consortium of about 150 companies, representing diverse industries such as automotive, consumer packaged goods, finance, government, insurance, media, telecommunications, travel and other computer industries. It is chattered to accomplish the following objectives:

- Promote web services interoperability across platforms, operating systems, and programming languages with the use of generic protocols for interoperable exchange of messages between services.
- Encourage web services adoption
- Accelerate deployment by providing guidance, best practices and other resources for developing interoperable web services.

Standard bodies that are also active in Web services include; World Wide Web Consortium (W3C), Organization for the Advancement of Structured Information Standards (OASIS), Internet Engineering Task Force (IETF), Java Community Process (JCP), and Object Management Group (OMG) [5]. WS-I as standards integrator supports the relationships with standards bodies who own specifications and fosters communication and cooperation with industry consortia and other organizations. It has a set of deliverables to assist in the development and deployment of Web services, including its profile of interoperable Web services. A profile is defined in the WS-I glossary as "a collection of requirements to support interoperability" [23]. WS-I will deliver a collection of profiles that support technical requirements and specifications to achieve interoperable Web services which includes the following deliverables:

- **Profile specification**

This includes a list of non-proprietary web services-related specifications at certain version levels, plus a list of clarifications and restrictions on those specifications to facilitate the development of interoperable web services.

- **Use Cases and Usage Scenarios**

These capture the business and technical requirements, respectively for the use of web services. These requirements reflect the classes of real-world requirements supporting web services solutions, and provide a frame work to demonstrate the guidelines described in WS-I profiles.

- **Sample Applications**

These demonstrate the implementation of applications that are built from web services usage scenarios and use cases, which conform to a given set of profiles. Implementations of the same sample application on multiple platforms using different languages and development tools, allows WS-I to demonstrate interoperability in action and to provide readily useable resources for the web services practitioners.

- **Testing Tools**

These are used to monitor and analyze interactions with a web service to determine whether or not the messages exchanged conform to WS-I profile guidelines.

5.4 Web Services and SOA

The first point to make is that implementing Web services is not equivalent to implementing an SOA. The service layer is only one tier of any potential SOA and successfully implementing one or more business functions as Web services does not necessarily guarantee subsequent success with a full-blown SOA. There are other equally vital layers within SOA which have to be taken into account including application and business process as well as the service layer. Indeed a technology centered approach to SOA will be unlikely to yield the best results, and it can be better to view a vendor's middleware product as part of a logical infrastructure service sub-layer in the SOA rather than the full story on its own [28]. It is also important to point out that web services are not the only technology that can be used to implement service oriented architecture. There are many examples of organizations who have successfully implemented service oriented architecture using other technologies. A well-designed, service-oriented environment can simplify and streamline many aspects of information technology if properly carried out. The technology set introduced by XML and Web services is diversely complex. Too often, organizations investing in Web services discover the errors of their ways once entire solutions have been built and deployed because most corporate IT departments do not employ any form of planned integration or migration strategy [3]. Strategizing with a foreknowledge of how to best incorporate XML, Web services, and service-oriented design principles into various domains of an automated enterprise, is a path which at the end lie a sophisticated and adaptive automation environment.

- **Reasons to consider Web services**

- The global IT industry is embracing and supporting Web services. By incorporating them early, organizations will gain an understanding of an important platform shift that affects application architecture and technology.
- The use of Web services does not require entirely new application architecture because of the loosely coupled design feature that allows addition of a modest amount of simple services, without much impact on the rest of the application.
- Incorporating service-oriented paradigms for organizations considering or already using a service-oriented design or business model can motivate the technical migration to a Web services framework.
- Most of the current development tools available today already support the creation of Web services, and these tools shield the developer from the low-level implementation details. This aids its learning and allows for a faster adoption of Web service-related technologies.

- **Basic Web Services**

The basic (point-to-point SOAP/HTTP) Web services provide a solid foundation for implementing a service-oriented architecture, but there are important considerations that affects their flexibility and maintainability in enterprise-scale architectures. First, the point-to-point nature of basic web services means that service consumer needs to be modified whenever the service provider interface changes. This is often not a problem on a small scale, but in large enterprises it could mean changes to many client applications. It can also become increasingly difficult to make such changes to legacy client. In the same vein, you can end up with an architecture that is fragile and inflexible when large numbers of service consumers and providers communicate using point-to-point “spaghetti” style connections [23]. Also, basic web services require that each consumer has a suitable protocol adapter for

each provider it needs to use. Having to deploy multiple protocol adapters across many client applications add to cost and maintainability issues.

5.5 Enterprise Service Bus

It is evident that Web services based technologies are becoming more widely used in enterprise application development and integration. One of the critical issues arising now is finding more efficient and effective ways of designing, developing, and deploying Web services based business systems. More importantly is moving beyond the basic point-to-point Web services communications to broader application of these technologies to enterprise-level business processes. In this framework, the Enterprise Service Bus (ESB) model is emerging as a major step forward in the evolution of Web services and service oriented architecture [23]. The Enterprise Service Bus approach is designed to cater for the shortcomings of the basic point-to-point Web service approach. The Enterprise Service Bus concept is not a product, but an architectural best practice for implementing a service-oriented architecture. It establishes an enterprise-class messaging bus that combines messaging infrastructure with message transformation and content-based routing in a layer of integration logic between service consumers and providers [23].

The main aim of the Enterprise Service Bus is to provide virtualization of the enterprise resources, allowing the business logic of the enterprise to be developed and managed independently of the infrastructure, network, and provision of those business services. Resources in the ESB are modeled as services that offer one or more business operations. Implementing an Enterprise Service Bus requires an integrated set of middleware services that support the following architecture styles:

- Service Oriented Architectures: where distributed applications are composed of granular re-useable services with well-defined, published, and standards-compliant interfaces
- Message-driven Architectures: where applications send messages through the ESB to receiving applications
- Event-driven Architectures: where applications generate and consume messages independently of one another

Also, the middleware services provided by an Enterprise service Bus needs to include the following:

- Communication middleware supporting a variety of communication paradigms (such as synchronous, asynchronous, request/reply, one-way, call-back), qualities of service (such as security, guaranteed delivery, performance, transactional), APIs, platforms, and standard protocols. s
- A mechanism for injecting intelligent processing of in-flight service requests and responses within the network
- Standard-based tools for enabling rapid integration of services
- Management systems for loosely-coupled applications and their interactions

An enterprise service bus aims at enabling a business to make use of a comprehensive, flexible and consistent approach to integration while also reducing the complexity of the applications being integrated.

5.6 Service Level Agreement

The business environment is highly competitive and the quality and guarantee of service is one of the substantial aspects for differentiating between similar service providers. A Service Level Agreement (SLA) between a service provider and its customers will assure customers that they can get the service they pay for and will obligate the service provider to achieve its service promises[21]. A service level agreement is an agreement regarding the guarantees of a web service which defines mutual understandings and expectations of a service between the service provider and service consumers [24]. The service guarantees are about what transactions need to be executed and how well they should be executed. Edward of Sun professional Services points out that a good SLA should address five key aspects:

- What the provider is promising.
- How the provider will deliver on those promises.
- Who will measure delivery, and how.
- What happens if the provider fails to deliver as promised?
- How the SLA will change over time.

In the description of an SLA, realistic and measurable commitments are important. Performing to expectations is very important as well as quick and well communicated resolution of issues. One of the challenges for a new service and its associated SLA is that there is a direct relationship between the architecture and what the maximum levels of availability are, which implies that an SLA cannot be created in isolation. An SLA must be defined with the associated infrastructure in mind. An exponential relationship exists between the levels of availability and the related cost [32]. Some customers need higher levels of availability and are ready to pay more which promotes the approach of having different SLAs with different associated costs. A service level agreement is very important to gain the commitment of the customers, sets key performance indicators for customer service and the internal organization. By establishing the price of non-conformance in form of penalties, the customer understands that the service provider truly believes in its ability to achieve the set performance levels which makes the relationship clear and positive.

- **Weaknesses of Web Services**

Though conventional wisdom invariably ties Web services to SOA, the usage however has some drawbacks especially in the areas of interoperability and security. Among which are extra overhead, performance implications, replay attacks, spoofing, message interception, denial of service attacks, service level agreement, evolving standards like WSDL, person in the middle attacks and so on. For Web services as with any application, it is necessary to establish the right balance between business requirements, protection, performance, and ease of administration.

6 THE GUIDELINE PROCESS

Johnson and Scholes defined strategy as the direction and scope of an organization over the long-term which achieves advantage for the organization through its configuration of resources within a challenging environment, to meet the needs of markets and to fulfil stakeholder expectations [14]. The word strategy is been used regularly among organizations both within the private and the public sector and is no longer strange that successes and failures of organizations are now attributed to its effective formulation. Despite the widely acclaimed benefits attributed to the implementation of SOA, not many organizations have made the move to adopt SOA. In a survey conducted among chief executive officers, chief information officers and IT managers of organizations involved in the development of service-oriented architecture, over 80 percent of the organizations replied that SOA was a strategic idea for their organization, but 56 percent stated that a well-defined SOA strategy is not yet in place [29]. This revealed the fact that making the move to service-oriented infrastructure has some major challenges associated with it and is not an initiative to be taken lightly.

Developing SOA platform constitutes a foundation upon which to launch more technically sophisticated move towards changing business requirements. Since it entailed the foundations that support the business processes, data, application and security, developing a clear strategy is more than critical. Successfully adopting a SOA will be dependent on cautious planning around the architecture's entire life cycle, starting from the development of services through deployment, and efficient management. Therefore to successfully make legacy systems to be service-oriented architecture compliant, there is need for a thought through implementation steps. After careful study and analysis of organizations that are planning to implement SOA coupled with those that have implemented it, the following steps can be applied as a guide to implement SOA most especially when migrating from legacy systems through the use of Web services.

6.1 The Interview

For this research to be more balanced and to get another perspective of service-oriented architecture and the use of Web services apart from the academic literature sources, an interview was conducted with a professional working on SOA. The person interviewed is a consulting architect presently working on an SOA project "automating the claims handling process among many others. The interview process lasted several days based on appointments until I was able to get all the necessary information needed to support this report. During this process, the information gathered from other sources such as academic literatures, professional journals and magazines were compared to see similarities and differences among them and establish a common ground for analysis. From the consultant's point of view, service-oriented architecture is a fantastic approach to building flexible systems, align business processes and realize tremendous benefits both in terms of business and IT. He pointed out that adopting SOA is essential to deliver the business agility and IT flexibility promised by Web Services. But these benefits are delivered not just by viewing service architecture from a technology perspective and the adoption of Web Service protocols, but require the creation of a service-oriented environment that is based on the following key points [34]:

- Service-oriented architecture is not just architecture of services seen from a technological point of view, but includes the policies, practices, and frameworks by which to ensure the right services are provided and consumed.
- For service-oriented architecture, it is better to implement processes that ensure that there are at least two different and separate processes both for the provider and the consumer.
- Service is very important concept and Web Services are the set of protocols by which services can be published, discovered and used in a technology that is not platform dependent and in standard form. Though other alternatives exists.
- Developers should consider the business perspectives as the starting point rather than discovering individual services and putting them into context.

By defining and delivering IT in terms of business services, the business/IT alignment will become easier to achieve because a business service binds the IT and business at the process level right into the functionality delivered.

Some of the reasons why his organization has embarked on service-oriented architecture projects are that the business organization is such that is located in different countries across the globe with different systems and solutions in these countries in different business areas. But to meet up with present business demands, there is need for one common solution. It is very hard for the different business units and branches of the organization to work in isolation; there must be cooperation and interaction among them to achieve the overall objective of the organization. Also, since the business logic is coded into these old systems, it becomes very hard to manipulate them to support the business needs, consumes a lot of time and very expensive to maintain and even change [34]. He emphasized the need to understand the IT environment and business processes before taking the crucial step to implement it. This is because; implementing a service-oriented architecture is a corporate decision that will have an impact on an organization for many years to come. The first important fact to recognize regarding SOA is that since it is an architectural model, it may take several years for an organization to fully implement its SOA vision.

- **Look Before Leaping**

Though the benefits of SOA are very enticing, many organizations jumped into this enormous undertaking before fully considering the implications it may have on their existing systems. Developing a good plan and strategy that includes a reasonable implementation timeframe is appropriate before moving towards SOA implementation [34]. It is better to spend a lot of time during the planning stage than to get into the middle of the implementation and get to a fixed point. At this stage, a lot of resources would have been wasted which can jeopardize the entire effort and threaten the stability of the organization. A good plan should include a definition of the organization's IT architecture, topology and road map for implementation, process for selecting hardware and software suppliers based on the needs of your organization. One of the most critical parts of the plan is to examine which components can be used in the newly defined SOA strategy, a part many organizations takes lightly only to pay the price later in loss of resources and time. A good knowledge of tools and application portfolio management tools can provide you with a broad list of databases, programs, files and third party software products and how they interact with each other. By having a good plan from the beginning, an organization will be able to determine whether it will undertake in-house development, license new package solutions, and redevelop applications [34]. The organization will know whether it has the capability to reuse existing systems or pursue any combination of these modernization options.

- **How to Start**

There is this issue across industry as to whether SOA works best when rolled out through smaller, targeted projects or as part of a broader enterprise effort. From the practical point of view, incremental approaches to integration have been the way forward for a number of years now and it is not surprising that organizations are starting SOA initiative on a small scale [34]. However, it is also important that the SOA initiative is not strangled at the beginning by skimping on the SOA ecosystem until more projects are implemented. It is better to think globally in terms of the entire SOA vision but act locally at the beginning. Building a business case for SOA could be tricky, because immediate cost savings might not be easily linked to larger projects. By starting small, an organization can learn how to monitor the benefits and see how their business is becoming more flexible and adaptable [34]. A small, manageable project can demonstrate how SOA principles reduce time to market, building support for a larger implementation. Using one project as a proof of concept, establishing the tools, methodologies, architectures, etc, and building the first set of reusable services, subsequent projects will be able to re-use these assets and artifacts and add new ones.

- **Technical Approach**

There are several technical approaches when it comes to implementing service-oriented architecture. Some of which are Service Component Architecture (SCA), Enterprise JavaBeans (EJB), Web services among others. But the most widely used approach is the use of Web services. Web services is a term used to describe a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone [34]. XML is used to label the data, SOAP is used to transport the data, WSDL is used for expressing the services available and UDDI is used for listing what services are available. It is basically used 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. It shares business logic, data and processes through a programmatic interface across a network and also allows different applications from different sources to communicate with each other without time-consuming custom coding. Because all communication is in XML, Web services are not fixed to any programming language or operating systems [34]. Web services aid the reuse of application components as services and connect existing software by solving the interoperability problem. However in some cases, the combination of two technical approaches might be considered depending on the expertise available within the organization.

- **Legacy Systems**

To implement a service-oriented architecture, an organization does not necessarily have to discard their existing system. A legacy system can be a computer system or application program that is still used because of the prohibitive cost of replacing or redesigning it, despite its poor competitiveness and compatibility with modern counterparts [34]. They are invaluable assets with embedded business logic representing many years of coding, developments, modifications and enhancements. Mostly these systems were developed independently without a consistent underlying architecture, resulting in overlapping and redundant functionality and data. Therefore a better approach is to find a way of modernizing these systems to remain competitive. However, modernized IT solutions must create new value from existing systems and provide flexibility and easy interoperability among a broad set of technologies which is usually a problem with legacy applications. Service-oriented architecture offers a practical solution for evolving and reusing existing assets. Transforming a legacy system requires a deep functional knowledge of the system [34]. It is vital to understand the business processes that are automated by the legacy system and to compare

those with the current needs with the aim of improving the outdated parts to create new value from existing assets. Since the legacy systems are based on outdated technological platforms that are often difficult to maintain and possibly not supported anymore, the target architecture must be based on leading-edge technologies, maximizing the reuse of existing assets.

One of the core activities of transforming legacy systems to service-oriented systems is to separate the business logic and application data. The methods or approaches that will be used in converting a legacy system will largely depend on how well componentized the legacy system is [34]. If the system is well componentized, then the wrapper approach can be used which encapsulates the existing business logic and reuse it with or without changes. But if otherwise, the system has to be repurposed into manageable pieces of functionality. There are different ways of accessing the functionality in a legacy system both when it comes to the technical connectivity and the access to business logic and data. On the technical side, is the need for some ways of accessing the data, this might be a CICS transaction gateway or similar, on top of which you can build a web service wrapper [34]. If the legacy system has native Web service capability, then there is possibility of exposing functionality directly as Web services. When it comes to access to business logic and/or data, it may get more difficult since the legacy system may or may not be built in a fashion that allows access to the functionality required. One way is to write a Web service wrapping some existing functionality or do some restructuring in the legacy system in order to be able to expose the requested functionality as a service. Generally speaking, the method to be used in order to find out how to use a legacy system in an SOA is to first define the desired functionality based on business processes to be implemented (top-down approach) [34]. Then analysis of the capabilities of the legacy system is done, followed by defining what functionality can be exposed and at which cost (man-hours, money, time; bottom-up approach). Most likely, a compromise between the top-down and the bottom-up service definitions will be adopted in order to both satisfy the business requirements and not drive up implementation costs too high.

6.2 Formulating the Guideline

Every great journey starts with a goal or destination which guides the activities at every stage of the journey. While service-oriented architecture is a powerful proposition, the work required to achieve the desired results must not be underestimated as with previous IT models. There is need to recognize SOA concept as a long term strategy rather than a development project that must be completed within one or two years. Despite the widely acclaimed benefit of SOA, most organizations are yet to embark upon it due to the complexity that surrounds its adoption. Many projects especially IT projects have failed as a result of lack of proper planning before the project kicked off. A fundamental approach to successfully adopt SOA is to have a good strategy put in place before the initial kick off of the entire project. To be successful, organizations must assess their strengths and weaknesses, establish clear direction, choose a route, and then consistently reassess that route as they follow through it. For this reason, this report presents a guideline that can be used by organizations when introducing SOA especially through the Web services technical approach. This guideline was a result of analysis by comparing the information provided by the academic literatures, scientific publications and that obtained from professional point of view through interview. Like I observed at the beginning of this report, the result presented here is limited in way as a result of the access to some companies within Sweden which could have given me a chance to compare several professional opinions. Nevertheless, the guideline is very much relevant for its intended purpose.

The following points are considered very important to be considered when introducing service-oriented architecture:

- **Recognize when to use Web services**

Business organizations should not be consumed by the perceived simplicity of Web services because the dynamics of developing and operating in an actual distributed, loosely coupled setting is an area that is yet to gain a lot of professionalism and expertise. A good starting point for a successful integration of Web services is for an organization to evaluate the true need for it. Irrespective of the wide acceptance, adoption and implementation of Web services in the IT mainstream, organizations should only incorporate it when it is evident it will add value to them. For instance, incorporating Web services may simply not be a requirement for autonomous application environments, such service becomes more important when taking interoperability requirements into account [34]. If the need has been convincingly established, the extent to which the Web services will be used must also be defined before they are developed. "Services can be phased in at different levels, allowing the organization to customize an adoption strategy." [30]. For example, starting with a single application project provides a low-risk opportunity and enables its integration to a limited extent and in a controlled manner. Although Web services interface is quite cheaper than a proprietary solution, it is a good idea to spend reasonable amount of time at the planning stages. This is because once a system is deployed, it becomes very difficult and costly to redesign.

- **Gain Sufficient Knowledge of Web Services Technology**

Knowledge and information are two practical phenomena that often lead to failure in any aspect of life. Although the concept behind Web services has a lot in common with traditional components-based design, it is still significantly different. Business organizations must have personnel who know the fundamental concepts behind this technology, strengths and weaknesses, its future applicability and industry standards and bodies [34]. Adding improperly designed Web services to an application may result into redeveloping them sooner than might be expected. When implementing a service-oriented architecture with this technology, it is recommended to limit the scope of Web services in the production environment to the scope of the available knowledge and information. For example, some tools and proprietary extensions will create dependencies on a vendor-specific platform [30]. The long-term implications of these extensions need to be fully understood before committing too many applications that will rely on them so that opportunities for future interoperability may not be compromised. As knowledge and expertise increases in the organization, further enhancement to the development of service interfaces can be carried out with better implementation of new services. At the initial stage of development, professionals and experts can also be consulted. Initial projects implementing Web services should be limited to low-risk prototypes and pilot applications until an adequate understanding of how Web services are best utilized within a technical environment.

- **Start Implementation with Transition Architecture:**

In principle, business ethics demand that a low investment or financial commitment be made when entering into a new market area, developing product prototype or any business initiative. This same idea can be applied when implementing service oriented architecture based on Web services form legacy systems. A cost effective way to gain experience with service-oriented technologies and concepts is to start the transition by delivering applications the normal way and then simply adding application proxies, or a custom designed frontage to the functionality that is to be exposed through a service interface [34]. This implies a

transition architecture that only introduces service-oriented concepts, without the technology itself [30]. The benefit of this approach is that it would be easier to quickly revert back to the traditional component-based model incase of any failure or incompatibility without too much impact to the overall application design. During this transition, a feasibility analysis can be conducted to measure the pros and cons of the Web services platform as they relate to development projects and technical application environment within business organization. By creating a showcase project, it will be proved that the chosen approach and technology is successful or otherwise.

- **Make Use of the Legacy System**

There are often very good reasons to replace or renew legacy environments in order to bring them into a contemporary framework such as flexibility and high maintenance cost. However the huge financial investment envisaged and the perceived high risk associated with modernizing legacy systems have prevented the upgrade. This is because legacy systems run critical business processes which make them to be business critical at any point in time. According to the free on-line dictionary of computing, Legacy system is a computer system or application program which continues to be used because of the prohibitive cost of replacing or redesigning it and despite its poor competitiveness and compatibility with modern equivalents. Legacy software and applications are often considered to be a financial base for an enterprise generating remarkable high profit margins, thus responsible for large amount of an organization's operating profit [34]. Therefore they must be rebuilt in such a way that ensures enterprise-wide business operations remain robust and compliant. Since legacy systems provide the most value-generating business processes, it is a good practice to always consider reusing legacy logic before replacing it.

Service-oriented integration architecture most often provides an important alternative of building on the legacy systems and bring application logic embedded in them into integrated enterprise. Through the use of adapters and a service interface layer properly designed for functional abstraction, Web services can help to take advantage of the legacy system [30]. Leveraging existing systems is extremely cost-effective, and the process of integration can be relatively expedient when compared to replacing a legacy environment altogether. Though this process can immediately broaden the resources shared by an enterprise, there are challenges with bringing previously isolated applications into interoperability loop.

- **Legacy System Reuse Approaches**

The approach of modernizing a legacy system can be divided into two broad categories.

- **Legacy Integration and Service Enablement**

Legacy integration and service enablement is a tactical approach to align legacy systems to business needs through noninvasive wrapping of legacy assets by using new layers of flexible technologies [8]. Technologies such as EAI solutions and messaging tools can be used and recently with standardized interfaces using Web services.

- **Legacy Transformation**

Legacy transformation is a strategic approach that aims to revitalize and streamline legacy systems to ease maintenance and extensions through invasive reengineering to augment the legacy systems architecture [8]. It involves a profound and detailed analysis of the existing code base, and an understanding of the system functionality and data architecture. Subsequently, it involves the extraction and rationalization of data definitions, data, and business rules. This is followed by an iterative process that involves refactoring,

consolidation, componentization, and redesigning activities to make the code more modular and to ease the incremental migration to a flexible architecture.

From the two named approaches, the legacy wrapping or service enablement can prove to be a very practical short-term solution because it promises quick turnaround and minimal changes to the existing legacy platform. When it comes to migrating from legacy systems into an SOA based system, the effort applied can vary widely depending on how well componentized the legacy system in question is. Most legacy systems are not componentized i.e. the business logic and data are not separated from each other. Separating business logic from data has the benefits of flexibility, scalability and less work to do when there are changes to data access [34]. However, modern systems are often componentized, which allows for relatively simple wrappers to be built on them exposing a meaningful capability as a service. For legacy systems that are not componentized, business logic is scattered throughout different programs and applications which means that some parts of the system must at least be componentized before any meaningful services can be built on top of it. The option of which approach to use will depend largely on the legacy system capabilities.

Wrapper Approach

- Encapsulates the existing business logic
- Typically "reuse as-is" without changes
- May require adding new interfaces/adaptors

Componentized Approach

- Repurpose existing systems into manageable pieces of functionality
- Separate logic based on business domain
- Position systems to increase reuse, easy maintenance and reduce redundancy

Which ever approach employed to transform legacy system to service oriented architecture based system, the functional capacity boundaries around legacy applications must be defined and must not be integrated beyond these boundaries [30]. It is important to understand the limitation of a legacy foundation because exceeding it will severely tax a legacy environment that is not designed for external integration of such. The design of a service requires a solid knowledge of the business model within which it will operate, as well as the technologies upon which it will be built. Services that will form or intends to participate in a future SOA will need to be in alignment with the design strategy and accompanying standards that are part of the overall SOA implementation plan. The service-oriented solution plan must be designed in a way to accommodate its probable migration path. Most modernization legacy attempts have failed to gain funding because they are perceived to offer no short term benefits and increase in business potential though they deliver returns on the long term [31]. However, the foremost benefit of any service-oriented environments is the intrinsic potential for immediate and future interoperability. Based on this potential, application architectures can be designed for integration regardless of whether they will be integrating anything immediately. For example, local requestors can be facilitated as well as future remote requestors by providing both coarse- and fine-grained interfaces based on standard naming and service description conventions.

- **Know the Boundaries of the Development Platform**

As key industry standards continue to mature, so does the feature set required for Web services to become fully capable of representing and expressing sophisticated business logic in enterprise environments. Until the second generation of Web services specifications is fully evolved, there remains a rather volatile transition period in which many middleware and development platforms compensate for the absence or immaturity of these standards by

supplying solutions of their own [34]. To manage this situation, organizations should define and work within the boundaries of their development platform. Several platforms supplement core Web services standards with proprietary extensions which are based on draft versions of specifications expected to become industry standards [30]. However, the extent to which these standards are supported may vary. Adequate knowledge of how vendor-specific extensions are implemented and the likely dependencies they impose on the long term is essential before any business organization decides to invest in them. A commitment to use these extensions takes a careful thought because at some point in the future, there might be the need to migrate away from these extensions. Some development platform boundaries are too restrictive by providing extensions to Web services at the cost of requiring that all requestors of the service be built using the same technologies. This defeats the purpose of designing service-oriented applications and ties an organization to a platform that offers little more than traditional component-based environments.

- **Use Abstraction to Protect Legacy Endpoints from Changing**

With the application of service interface layer, it becomes easy to see how integrated legacy applications evolve. Configuration management procedures can be altered and improved by the use of abstraction. Since the integration layer acts as an intermediary between previously tightly bound legacy applications, a level of decoupling is achieved which makes the Web service the only contact point for legacy environments [34]. To illustrate this point, let us consider two integrated legacy applications A and B with only application A requiring an upgrade. With the above achievement, an upgrade exercise can be carried out on application A without requiring changes to application B. However, depending on the nature of the upgrade, application A's Web services may be affected and modifying the integration layer tends to be much easier than making changes to legacy logic. The level of abstraction that can be provided by service-oriented integration architecture can also significantly reduce the impact of platform migrations.

- **Build Around a Security Model**

Most often designers put together designs and sometimes build a preliminary version of an application without a serious thought for the underlying security model. A functional application design needs to be built upon the security model and not the other way round. Web services security models are unique, complex, and multidimensional. A key component of a standardized service-oriented enterprise is a service-oriented security model [30]. Web services security model can mean a number of different things, for instance a model can represent the security rules and technologies that apply to an application. This model can be implemented within a dedicated security services layer that also becomes an application architecture standard. The required use of a service oriented security (SOS) model can impose significant restrictions upon application designs. But this is however offset by the fact that its use also alleviates application development projects from having to deal with many of the issues relating to security but just to deliver an application that conforms to SOS standards.

- **Incorporate Web Service Standards**

As the world is moving towards a global village, standardization is one of the key factors emphasized to achieving global integration. Just with any enterprise application development project where you have different units of developers building different parts of the system, standardizing how each part is designed is important for all the traditional reasons like robustness and maintenance. The real benefits of deploying service-oriented architecture lie in establishing a standardized application interface [34]. This can potentially translate an enterprise into a standardized system for navigating application logic, integration

architectures, corporate data stores and parts of the enterprise infrastructure. Using Web services standards for the enterprise infrastructure will enhance efficiency and industry compliance. Developing services within an enterprise establishes infrastructure through which developers, integrators, and perhaps even business partners may need to navigate in the years to come and navigating through a non-standard environment will always slow down progress and introduce new risks. Building upon this is also to consider naming conventions as a means of labeling enterprise infrastructure [30]. When assembling the pieces of integration architecture, the end result is a multitude of interdependent components each a necessary link to the solution.

This creates a chain of inconsistent endpoint names especially when the environment consists of a mixture of legacy and contemporary application components.
The benefits of naming standards are often not evident until later in the project life cycle, when there is need to start plugging pieces together. At this stage, introducing a naming convention can become very inconvenient, messy and frustrating for teams working on the project. In some instances, the names used to identify public components and service interfaces acts as a reference points for other components and services [30]. Therefore when there is a change in a name in the middle or conclusion of a project, the change must also be effected to all the references to it. As a result, renaming all your solution's components and services turns into a major project itself, during which all further development is halted. Using a naming convention will not only improve the efficiency of administering solutions, it will make the migration and deployment of new integration projects much easier and reduce the risk of human error and the chance that a simple adjustment will lead solutions to total breakdown. Naming conventions should be easily accessible by every member of a project team, developers, administrators and anyone involved within the integration project.

- **Design Service Interface**

When modeling a service-oriented framework, there is need to provide a complete description of enterprise endpoints which makes the standardization of these descriptions very significant. Service interfaces should be described in a consistent pattern in order to establish a standard endpoint model which will result in a standardized service-oriented integration architecture that can be positioned as part of enterprise infrastructure. To ensure consistency in end point design, the common development process for a Web service needs to be reversed. Instead of building application logic and then expressing this functionality through an appropriate service interface, the design of the interface is carried out first. At this point, the naming conventions are incorporated with the enterprise-wide interface design standards. With a fundamental knowledge of what Web services will be encapsulating, a generic, consistent interface with operation characteristics that comply with a standard model can be created, and then the back-end logic built [30].

Organizations that are successful in interface design are found to have a resource responsible in order to ensure consistency across the interfaces within a Web services framework. Designing a Web service is a separate task from its actual development. A service interface designer is responsible for ensuring that the external interface of all Web services is consistent and clearly represents the service's intended business function. A step further is to classify and standardize service types using service models. Every Web service is unique, but many end up performing similar functions and exhibiting common characteristics allowing them to be categorized [34]. The use of service models can facilitate the application of specific design standards to different service models and the model type will instantly communicate a service's overall role and position within architecture. Service models can be used to gauge the performance requirements of service-oriented applications and models can be aligned with enterprise policies and security standards.

- **Remember Second Generation Specifications during the Design Process**

As more legacy application logic is expressed and represented within service-oriented environments, the demand increases for Web services to support a wider range of traditional business automation features. In response to these demands, the IT community improves or sometimes replaces technical specification. The feature set of the Web services framework continues to grow, driven both by standards organizations such as WS-I, W3C and major corporations like IBM and Microsoft which collaboratively produce specifications that address new functional areas for Web services to utilize. Therefore it is very important to approach the choice of each second-generation specification as a strategic decision point which will have implications on the architecture, technology platform, and design standards [30]. Regardless of whether it is part of the plan or not to incorporate the features offered by some of the newer second-generation Web service specifications, it is safer to design SOA with a foreknowledge of emerging specifications and their standards. Any of these specifications and standards that are considered significant or relevant can then be positioned within the future-state enterprise architecture. This can be achieved by keeping abreast of information on Web service standards.

- **Streamline Business Models Based on SOAs**

One of the most significant benefits of Web services technology platform is the provision of a more flexible, interoperable, and standardized model for hosting application functionalities. Service-oriented designs open up new opportunities for business automation and to take advantage of these opportunities, business models must be redesigned. A service-oriented architecture will increase the interoperability potentials between legacy systems which allow various business processes that rely on multiple applications or data sources to be reevaluated. With this platform, a collection of generic business and utility services will provide a number of ways to automate new parts of business units. Also, services can integrate with enterprise application integration solutions to deliver new business processes that in turn integrate existing business processes.

- **Use a Private Service Registry when Implementing Web Services**

Once Web services have been established as a common part of an enterprise, they will soon require interface upgrades and spawning new generations of services. Since some of them will always be in a state of transition, it will be difficult to keep track of many of the service interfaces. A good practice is to incorporate a private service registry to centralize published service descriptions into one accessible resource [30]. This acts as the central repository for current service interface information to which anyone interested will go to discover and learn about an enterprise's service framework. It aids efficient access to service interfaces, preserve the integrity of published service interface, and encourage discovery of generic and reusable services. For this service registry to be meaningful, its usage must be made mandatory and kept updated. If the use of this registry is a requirement that is strictly adhered to, it will become a core part of the administrative infrastructure supporting development projects as a resource center for published Web service end points. A good way to ensure that a registry is kept current and available is to assign ownership of these responsibilities like a Service Library Manager [3]. Maintaining a service registry is a unique job that requires uncommon combination of skills as the registry needs to provide a high level of availability and dependability. The Service Library Manager owns the organizations utility services and changes required to these services will need to be approved by the library manager. Such changes must be implemented in such a way that they are sufficiently generic for the future use and do not break existing interfaces already in use by service requestors. In a situation where an organization opens its registry to external business partners, the

manager is responsible for managing the authentication and authorization of users from outside the organizations.

- **Have an Effective Administration Strategy in Place**

An often overlooked aspect of projects implementing service-oriented applications are the subsequent maintenance tasks required to keep these environments going. Organizations should be prepared for the costs and complexities in administering a service-oriented enterprise. Increased interoperability results in a higher amount of dependencies between application environments, specifically their Web service endpoints. High level of integration comes with the responsibility of keeping Web services running smoothly, high usage volumes, error conditions, and other environmental variables need to be anticipated [30]. Administration costs should be properly represented in project budgets in other to guarantee the support infrastructure required by service-oriented architectures which can jeopardize the success of the application and those that are integrated with it. In an environment where many Web services are deployed and utilized, an administration system needs to be in place in order to ensure a reliable runtime hosting environment [34]. To achieve this, a Service administrator can be assigned who is responsible for the maintenance and monitoring of these hosting environments. Service administrators need to be proficient in the use of maintenance and monitoring tools in order to effectively handle unpredictable usage volumes and respond quickly to production issues relating to the availability and performance of Web services.

- **Monitor and Respond to Changes in the Service Hosting Environments**

Organizations need to be sensitive to changes and the rate of these changes and respond to them accordingly. When Web services represent the endpoints to existing or new integration channels, they can easily become the most demanding parts of an integrated environment. This can strain the underlying areas of the infrastructure supporting those services. Surveying existing infrastructure and identifying parts that may need upgrade is a good way to avoid performance bottleneck and to be responsive to infrastructure requirements. In preparing for any new application, there will be obvious parts where upgrades will definitely be necessary, like new servers, more memory, and routers are all typical requirements needed to support incoming application hosting environments.
Another salient point is that for organizations implementing service-oriented architecture to reap the benefits of interoperability promised by Web services, there is need to incorporate a multi-client test phase [30]. This precaution is especially important when working with second-generation specifications. A good approach to address this issue is to increase the amount of testing each newly created Web service will be subjected to and that test strategy should require that services be tested with a range of client's representative of potential service requestors. For example, a project team can create an application using J2EE, while another team based their application on .NET. Though neither team needs their application to integrate with the other, their respective testing phases should still include client requestors based on both J2EE and .NET.

- **Properly Organize Development Resources**

A common mistake in development projects that incorporate a limited SOA is to have one team deliver the Web services, and the remaining team(s) develops the balance of the application [30]. This sounds reasonable from the resource point of view because each team works with technologies that match their respective skills. This can create however a disconnection between developers responsible for building parts of an application that delivers a logical unit of business functionality. Development teams should be grouped

around logical business tasks. An application task should be broken down into its primary business functions which can be equivalent to a series of subprojects that collectively make up the application's feature set. Each of these subprojects will typically require the creation of a number of application components, some of which may be encapsulated within Web services. Development teams should be grouped in accordance with these subprojects, so that the performance and functionality of individual business tasks can be streamlined.

- **Organize Training for Developers**

In many cases, Web services affect the application's business model and the execution of an automated business task will differ within an application depending on how Web services are utilized. Developers must be unaware of the relationship between services been developed and its associated business tasks which is why it is critical to ensure that developers and designers are capable of applying the right blend of business and technical intelligence to every Web service [34]. If they are especially new to this platform, the importance of undergoing training cannot be overemphasized. For instance, recurring requirements for Web services are that they be generic, openly accessible, and relatively independent [30]. To achieve these distinctiveness, the functionality within a service needs to be carefully defined which requires knowledge of the business functions an application needs to deliver now, as well as the level of interoperability it will have to provide in the future. Generally, developers are always focused on immediate project requirements and creating Web services with contemporary development tools only gives a developer a false sense of confidence.

7 FINANCIAL CONSIDERATION

Service-oriented architecture is not only an architecture to build systems, but also an approach to make the business and IT cooperate in a more efficient manner in order to realize business benefits. Modernization of legacy systems proposed by IT departments in organizations which would have delivered tremendous business benefits failed as a result of lack or insufficient financial support from management. What looks like an obvious solution to business challenges to the IT department is not always seen in the same light by management. Human issues have played a strong role in what would otherwise be a clear business decision. Therefore IT departments planning to implement service-oriented architecture using Web services or any other technology must present their proposals to clearly reflect the justification of such financial investments. Web services are expensive and require a great deal of work to ensure that the benefits claimed are truly delivered. The budget must be thoroughly prepared to accommodate not only the immediate design, development and implementation cost but also maintenance expenses.

The financial implication for an organization depends on whether they are building an entirely new system or migrating from a legacy system. The cost of custom-developed services added to existing legacy environment will be typically lower than when starting the integration by investing in enterprise service-oriented middleware products. Putting together a realistic financial justification data on investment can be an enlightening experience especially for management to justify the use of Web services to implement service-oriented architectures and can also provide valuable information beyond that required to justify the project. In addition to providing evidence that predicted cost savings resulting from the use of Web services will be realized, properly researched financial data can give a clear idea as to how long it will take these benefits to be attained. The initial SOA migration may be the most expensive part of an enterprise-wide initiative which will reduce as the system gets matured over time.

7.1 Capital Investment Decisions

In an organization, every decision made that changes the organizations' cash flows relative to time can be considered as an investment decision. This attempt always aim at adding value to the organization or better position the organization for future challenges or outwit business competitors. Capital investment decisions are those decisions that involve current outlays in return for a stream of benefits in future years [27]. Often a decision to invest in a capital project involves a largely irreversible commitment of resources that is subject to a considerable degree of risk. Such decisions have consequential effects on organization's profitability and flexibility over the long term, thus requiring that they be part of a carefully developed strategy that is based on reliable appraisal and forecasting procedures.

Time remains the sole distinguishing feature between short-term decisions and capital investment or long-term decisions. Most short-term decisions reap the benefits of fund committed within a period of one year from the time of investment. Conversely for capital investment decisions, a significant period of time passes between the outlay and the recovering of the investment. For this reason, decisions on investment which take time to mature have to be based on the returns which that investment will make. Organizations, most especially equity financed i.e. organizations that execute projects by either issuing new ordinary shares or from retained earnings strive as much as possible to maximize the funds entrusted to them by their shareholders. Therefore in making investment decisions on capital

projects, they consider the opportunity cost of investing in that project to that of investing in securities in financial markets such as ordinary shares or government bonds.
An opportunity cost (also known as minimum required rate of return, cost of capital, discount rate or interest rate) is a cost that measures the opportunity and financial benefits that are forgone or sacrificed when the choice of one course of action requires that an alternative course of action be given up [27]. It represents the lost contribution to profits arising from the best use of the alternative forgone. When investing in that capital project is compared to investing in financial markets, the rate of returns that are available from investments in financial markets in securities with different levels of risk represents the opportunity cost of an investment in that capital project. This implies that if cash is invested in the capital project, it cannot be invested in another place to yield a return. An organization will therefore only invest in capital projects when there is evidence that they will yield a return in excess of the opportunity cost of the investment. For this reason, a proposal that incorporates the opportunity cost of an investment does not only stand a chance for success but also provides decision information for management. There are several investment criteria available to compare returns on an investment in a capital project with an alternative equal risk investment in securities traded in the financial markets. They are net present value, internal rate of return, payback method, and accounting rate of return or return on investment.

- **Net Present Value (NPV)**

Net present value is a way of determining whether a project yields a return in excess of the alternative equal risk investment in traded securities that takes into account the time value of money [27]. This can be calculated by finding the present value of the future net cash in flows of the project and subtracting the project's initial investment outlay using the formula below:

$$NPV = \sum_{i=1}^{n} \frac{FV_i}{(1+K)^i} - I_0$$

Where I_o represents the investment outlay and FV represents the future values received in years 1 to n. The rate of return K used is the return available on an equivalent risk security in the financial market. A positive NPV for a project provides an absolute value of the amount by which the project's investment exceeds the return available from an equivalent risk investment in securities traded in the financial market. Conversely, a negative value indicates the amount by which an investment fails to match an equal risk investment in financial securities. A positive NPV therefore indicates that an investment should be accepted while a negative NPV indicates that it will be rejected.

- **Internal Rate of Return (IRR)**

The internal rate of return represents the true interest rate earned on an investment over the course of its economic life. It is an alternative technique for use in making capital investment decisions that also takes into account the time value of money. The rate of return is the discount rate at which NPV equals zero [27]. The IRR is the interest rate K that when used to discount all cash flows resulting from an investment, will equate the present value of the cash receipts to the present value of the cash outlays. It can be calculated using the formula below:

$$I_0 = \sum_{i=1}^{n} \frac{FV_i}{(1+K)^i}$$

The IRR is calculated by ascertaining the discount rate that will cause the NPV of a project to be zero which can be found by trial and error by using a number of discount factors until the NPV equals zero. If the IRR is greater than the opportunity cost of capital, the investment is profitable and will yield a positive NPV. Alternatively, if the IRR is less than the cost of capital, the investment is unprofitable and will result in a negative NPV.

- **Payback Method**

This is one of the simplest and most frequently used methods by organizations to assess capital investment. A project's payback period is the length of time that is required for a stream of cash proceeds from the project's investment to recover the original cash outflow [7]. The payback period of an investment can be calculated by adding up the cash flows expected in successful years until the total is equal to the original outlay. Despite its wide usage, there are some shortcomings associated with payback method. First, it does not take into account cash flows that are earned after the payback period, and secondly, it ignores the time value of money. There are tendencies that using payback method for multiple projects will rank them incorrectly and can also results in the acceptance of projects that have a negative NPV.

- **Accounting Rate of Return (ARR)**

The accounting rate of return is also known as return on investment (ROI) or return on capital employed. It is different from other methods in the sense that profits rather than cash flows are used. It expresses the annual average profits arising from a project as a percentage return on the average investment required for the project. ARR is calculated by dividing the average annual profits estimated from a project by the average investment cost. Assuming that depreciation represents the only non-cash expense, profit is equivalent to cash flows minus the depreciation. The method used for calculating depreciation will determine the average investment figure that is used in the calculation of ARR.

$$ARR = \frac{average\_annual\_profits}{average\_investment}$$

Accounting rate of return is better when compared to payback method because it accommodates differences in the useful lives of the assets being compared. Nonetheless, it also ignores the time value of money. If the method is applied to a project where the cash inflows do not occur until the near end of its life, it will show the same ARR as it would for a project where the cash inflows occur early in its life providing that the average cash inflows are the same.

- **Making the Choice**

The first two techniques discussed above, the net present value and internal rate of return takes into account the time value of money and are very efficient and reliable. But the NPV is considered to be theoretically superior to IRR because of some drawbacks of IRR. For instance, using IRR method to rank mutually exclusive projects correctly cannot be guaranteed because of its reinvestment assumptions. Also, the returns generated by IRR method can be misleading when choosing between alternatives because it expresses the result as a percentage rather than in monetary terms. Though they do not take into account the time value of money, payback method and accounting rate of return are commonly used in practice. Payback method is frequently considered useful when organizations face liquidity constraints and require fast repayment of their investments [27]. ARR is a widely used financial accounting measure of managerial and organization performance. For this

reason, managers are likely to show interest in how any new investment contributes to the business unit's overall accounting rate of return.

The choice of which evaluation method to be employ by an organization will considerably vary, but should be based on the organization's business strategy and management decision policies. It is however strongly recommended that the net present value method should be given utmost consideration and used because of its reliability in most project assessments. I will also recommend to the IT department that the project proposal be prepared closely with the accounting or finance department to make it up to standard.

7.2 The Process

Calculating the return on investments for a SOA is not that simple and organizations will often have to wait several years before they can get a positive return off their investment. A good approach to this is to compare the return on investment that an organization would achieve normally by using a conventional integration approach to that of using a service-oriented approach. At the initial stage, it is expected that the return on investment will be negative for the SOA solution approach. This is due to investment made into a new architecture, tools, personnel training, consultation etc. This will be followed by an increase in return on investment over the following years until it reaches a break even point and eventually the return on investment will rise above that of the traditional integration solution when more and more applications and systems becomes service-oriented enabled and Web services interface compliant. When calculating the return on investment and making the business case for SOA, it is important to consider both the business benefits and the IT benefits in the calculation. For example, the following business and IT benefits can be emphasized but not limited to these:

IT Benefits

- Reduced cost of connection and maintenance
- Facilities to reuse existing assets
- Reduced complexity of integration
- Platform and technology independence
- Reduction in technology cost through commoditization
- Reduction or elimination of development effort to support new connections

Business Benefits

- increased business agility
- increased revenue and operational efficiency improvements
- improved customer satisfaction
- improved ROI of existing IT assets
- reduce time to market
- reduced vendor lock-in and switching costs

These benefits can be strategically analyzed by linking them with a practical situation to further expand their impact. For example, the improvements to a business process that have been enabled by a more flexible SOA architecture can lead to increased customer satisfaction leading to an increasing customer base and reduced costs for the call center. The portion of organizational fund normally allocated to service call center and improve customer satisfaction can then be channeled to another segment of the organization. In practice it might be difficult to have a perfect estimate for your financial estimates, therefore further

revisions to the financial justification data should be carried out as information gets updated to improve the accuracy of its predictions as they relate to subsequent phases in a long-term implementation program.

7.3 Future Study

Though Service-oriented architecture and Web services are growing quickly and their adoption is gradually increasing among organizations, there remain some challenging areas that need to be tackled. For example overcoming barriers to integration from a legacy system point of view, ensuring uniform data format and semantic consistency, industry supporting technologies (second-generation Web services applications) that are still developing, service level agreement, Web services security like replay attacks, denial of service attack, spoofing message interceptions e.t.c [1]. While organizations are still engulfed with the concept of SOA in general, talks have been going on about the next-generation version of SOA called SOA 2.0 or advanced SOA by industry leaders like Oracle and Gartner. SOA 2.0 is the term used to describe the combination of service-oriented architecture and event-driven architecture. Event-driven architecture is been described as the main distinction between SOA 2.0 and the first, client-server driven iteration of SOA in which deployments are based on request and reply [25]. Not all business processes and software topologies fit the first SOA model. With SOA 2.0, an event-driven architecture is deployed in which software modules are related to business components, and alerts and event notifications are featured. In request-reply SOA, a service retrieves information or performs an action on behalf of the requester to produce a result. Event-driven SOA has the sources, or initiators of activity, notify the environment of a change and the execution code that processes the notification at some point, possibly after additional events are detected.

Among these challenging areas in SOA and Web services, an interesting area of possibility for future study is the Web Services Service Level Agreement e.g. performance optimization. IT has moved beyond the stage of Web service availability and deployment with respect to a given service, there must be assurance that the Web services SLA for that service is being met [21]. Different requirements apply to services as they begin to be consumed by multiple applications and users. Issues to be considered are for example, ensuring the service level requirements of consumers are being met, understanding the pattern of usage of services. A good understanding of how services are actually used is necessary in order to optimize services, resolve issues and detect anomalies that might likely occur. Information about consumers and their use of services can assist to understand such things as when the services network is approaching the limits of its capacity and additional capacity will be required [21]. Also operations performed by consumers and the time of this operation and which consumers, if any, are causing problems for others. Service Level Agreement can be used to manage the performance of a service, improve the quality of the development process, reduce the risks of project failure and strengthen customer relationships.

8 CONCLUSIONS

Service-oriented architecture is indeed a way to strategically align IT and business especially when implemented through the use of Web services. Web services remain a powerful tool as a result of the standards that underpin this technology and the unprecedented level of support these standards enjoy. Based on widely adopted industry standards such as eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), and Web Services Description Language (WSDL), Web services can easily and efficiently put together different applications written in different programming languages and running on different operating systems and hardware, while using universally available internet protocols such as Hypertext Transfer Protocol (HTTP). Whether an organization is running on Unix or Windows operating systems or uses Java or C programming language to design applications, the stage is set to enable them implement a SOA based system. As a result of the world wide development and standard bodies involved in Web services, every business sector stands to benefits from its application. Organizations only need to keep it simple, keep it incremental and continue to learn as much as possible.

A success tip is that it is more appropriate to start the SOA activities with the business objectives rather than leading with SOA IT solutions which is likely to be the case with many organizations and ensure the availability of business experts during the planning stages. The planning stage is very important for the overall success of the project which can have a negative effect on other stages if not carefully carried out. SOA is actually about business improvement, hence business analyst are critical to its success. It is essential to consider hiring external consultants who are experts and also the option of outsourcing some aspects of the SOA and Web services implementation. There is no doubt about the benefits an organization will derive from implementing a SOA both from the technical and the business perspective. Every business has its own rules that must be followed to achieve success, so also is with technology and its implementation. Since legacy systems are critical to the success of business, a well thought process of upgrading them to meet up with current business requirement and technological performance is what business organizations should considered. The methods or approaches that will be used in converting a legacy system will largely depend on how well componentized the legacy system is. If the system is well componentized, then the wrapper approach can be used which encapsulates the existing business logic and reuse it with or without changes. But if otherwise, the system has to be repurposed into manageable pieces of functionality. The method to be used in order to find out how to use a legacy system in an SOA is to first define the desired functionality based on business processes to be implemented. Then analysis of the capabilities of the legacy system is done, followed by defining what functionality can be exposed and at which cost.

Since service-oriented architecture adoption requires a huge financial investment, having a good financial plan and return on investment is considered a good approach from the management's perspective. Organizations' top management executives needs some form of financial return on investment in order to make decisions whether to embark on that kind of capital intensive project. From the above, it is clear that any organization that is in the process of adopting service-oriented architecture must have a good understanding of the big picture of SOA concept. An organization that successfully deploys a wide-spread SOA will find itself been able to respond to market opportunities and competitive threats with a speed and agility that is not possible for organizations still constrained by more traditional IT implementations. There is no doubt that SOA will become a main-stream architectural approach within a couple of years. With adequate planning and a good implementation guide, organizations can be optimistic of reaping the desired business and IT benefits of adopting service-oriented architecture.

9 ACKNOWLEDGEMENTS

I have acquired valuable knowledge about Service-oriented architecture and Web services during the compilation of this thesis work. Though challenges lie ahead in this area of IT and business technology, I will be willing in the future to carry out detailed and further study in a specific aspect of implementation using these technologies. I give glory to God Almighty the source of life. I will like to express my profound gratitude to my supervisor Jenny Lundberg who has relentlessly monitored the success of my work and offered valuable advices which has assisted me in unquantifiable ways. Thanks for your time, patience, love, motivation and inspiration for success. To Pascal Laue with Accenture AB, your professional advice and time during the interview process cannot but be mentioned. I also appreciate Rune Gustavsson for his constructive criticism and guidance.

References

[1] Alesso H., Peter Smith, Craig F. Developing Semantic Web Services. A K Peters Limited, 2004.

[2] Andrew Schwarz and Rudy Hirschheim. Service Oriented Governance: Key Principles for Strategic Success. Cutter Benchmark Review on Analyzing IT Metrics for Informed Management Decision, Volume 6, Number 10 October 2006.

[3] Arora Geetanjali, Kishore Sai. Building Web Services with XML. Premier Press Incorporated, 2002.

[4] Bisbal J., Lawless D., Wu B., and Grimson, J. 1999. Legacy Information Systems: Issues and Directions. IEEE Software, vol. 16, no. 5, pp. 103-111.

[5] C. Matthew MacKenzie etal. Reference Model for Service Oriented Architecture 1.0, Committee Specification 1, OASIS 2 August 2006.

[6] Cimitile A., Lucia A. D., Lucca A. D., and Fasolino A. 1997. Identifying Objects in Legacy Systems. 5th Workshop on Program Comprehension (WPC '97), Los Alamitos, CA, pp. 138-146.

[7] Colin Drury. Management Accounting for Business, 3rd Edition. Thomson learning, 2006.

[8] Deursen A. v., Elsinga B., Klint P., and Tolido R. From Legacy to Component: Software Renovation in Three Steps. 2000.

[9] Edward Wustenhoff. Service Level Agreement in the Data Center –Sun Professional Services, Sun Blueprints Online April 2002

[10] Eric Newcomer, Greg Lomow. Understanding SOA with Web Services. Addison-Wesley, 2005

[11] Ferris C., Farrell J., What are Web services? Communications of the AMC, Vol. 46 Issue 6, June 2003, p. 31

[12] Gabriele Piccoli. Service Oriented Architecture: Old Wine in New Bottles or a Critical Tool for the Organization? Cutter Benchmark Review on Analyzing IT Metrics for Informed Management Decision, Volume 6, Number 10 October 2006.

[13] Gabriele Piccoli. Another Bottle of Wine, Same Risk of Hangover. Cutter Benchmark Review on Analyzing IT Metrics for Informed Management Decision, Volume 6, Number 10 October 2006.

[14] Gerry Johnson, Kevan Scholes. Exploring Corporate Strategy, 6th Edition. Financial Times/Prentice Hall, 2001

[15] Grigoris Antoniou, and Frank Van Harmelen. A semantic Web Primer. The MIT Press Cambridge, Massachusetts, 2004.

[16] Hans Brunner. Service-Based Architecture Best's Review; Dec 2003; 104, 8; ABI/INFORM Global pg. 96

[17] JPMorgan, Schwab, Ian Bruce. Prove Web Services Is More Than Just Hype, Wall Street & Technology; Apr 2004; ABI/INFORM Global pg. 48

[18] Jussi Koskinen, Software Maintenance Costs. Information Technology Research Institute, ELTIS- Project University of Jyväskylä 2003

[19] Lauren Bielski, SOA Begins to Make Inroads American Bankers Association, ABA Banking Journal; Dec 2006; 98, 12; ABI/INFORM Global pg. 48

[20] L. Walker. IBM Business Transformation Enabled by Service Oriented Architecture IBM System Journal, Vol 46, No 4, 2007

[21] Li-jie Jin, Vijay Machiraju, Akhil Sahai. Analysis on Service Level Agreement of Web Services. Software Technology Laboratory, HP Laboratories Palo Alto HPL-2002-180 June 21st, 2002

[22] Michael Rosen. Adoption of Best Practices in Service Oriented Architecture Development. Cutter Benchmark Review on Analyzing IT Metrics for Informed Management Decision, Volume 6, Number 10 October 2006.

[23] Mark Endrel, et al. Patterns: Service Oriented Architecture and Web Services. IBM Red Books, IBM 2004.

[24] Marshall Breeding, Introduction to Web Services. Library Technology Reports; May/Jun 2006; 42, 3; Academic Research Library pg. 5

[25] Murthy Nukala, M. Rangaswami, Tom Stein. Getting Your Arms Around Web Services, Optimize. ABI/INFORM Global pg. 55 March 2003

[26] Paul Gray. SOA-Moving From Doing it Yourself to Getting Others to Do it For You. Information Systems Management 2007; 24, 1; ABI/INFORM Global pg. 98

[27] Richard Brealey, Stewart Myers, Alan Marcus. Fundamentals of Corporate Finance, Fifth Edition. McGraw-Hill Irwin 2007

[28] Sprott D. L Wilkes. Enterprise Framework for SOA. CBDI Journal, 17 March 2005.

[29] Survey of over 200 CEOs, CIOs, and IT managers EbizQ, Q3 2005.

[30] Thomas Erl. Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. Prentice Hall, 2004

[31] Toby Sucharov. Mainframe Makeovers. Information Professional, Institute of Engineering and Technology, Volume 4, Number 6 Dec/Jan 2007/08.

[32] Zakaria Maamar, Djamal Benslimane, Nanjangud C., Narendra. What Can Context Do For Web Services? Communications of the Association for Computing Machinery, volume 49, Number 12, December 2006.

[33] Ziff Davis. Strategies for SOA Success, Ziff Davis Media Custom Publishing December 2005.

[34] Pascal Laue, Consulting Architect on SOA Project Automating the Claims Handling Process (Interview) 2007.

10 APPENDIX

Typical Questions on SOA

1. What is the name of your organization?

   Answer: *(Client)*

2. What is your type of business or company?

   Answer: *Property and casualty insurance*

3. What is your type of customers?

   Answer: *Private and commercial*

4. How many employees do you have?

   Answer: *Approx. 6600*

5. Do you have an IT department in your organization?

   Answer: *Yes*

6. What is your role or job in the organization?

   Answer:

   Consultant: Architect on an SOA project automating the claims handling process

7. What are the shortcomings of the present architectural systems used to support business in your organization?

   Answer:

   - *Different systems and solutions in different countries and business areas, one common solution required*

   - *Business logic coded into old mainframe systems, very hard, time-consuming, and expensive to change*

8. Has someone compared the current architecture with best practices in your industry to spot issues that need correction, such as the architecture's inability to align and keep up with the business?

   Answer: *Yes*

9. Do you have knowledge of what is called SOA?

   Answer: *Yes*

10. Is your organization currently working on implementing an SOA or plans to do it in the future?

Answer: *Yes*

11. If so, do you really think you need an SOA?

    Answer: *Yes*

12. If you do, how do you plan to implement it?

    Answer:

    Introduction of a tool set (IBM WebSphere). Use one project as a proof-of-concept, establishing the tools, methodologies, architecture, etc. and building the first set of reusable services. Subsequent projects will be able to re-use these assets and artifacts and add new ones.

13. Are you aware of several technologies that can be used to implement SOA?

    Answer: *Yes*

14. Can you name some of the techniques you know?

    Answer: *Web services, SCA*

15. Are you familiar with the concept of Web Services as a technique for implementing SOA?

    Answer: *Yes*

16. What is the difference between Services and Web services?

    Answer:

    A service is the observable set of behaviors [of a system] accessible via a prescribed interface (definition by OASIS) – this is a very generic definition of an interaction style. A web service is a specific type of service, describing the interface using the WSDL, using SOAP over HTTP as a transport protocol, etc.

17. Which technical approach do you prefer Web Services or other technique name it?

    Answer:

    I prefer to combine web services and SCA bindings, both have their advantages. Usually I would be using web services for everything exposed to the outside and SCA for internal bindings due to much better performance of the latter.

18. What is the difference between SOA and Web Services?

    Answer:

    A Service-oriented architecture is an architecture that defines how business functions implemented by independent systems work together to execute a business process. Web services are merely one technology of many (even if it probably is the most widely used one) to allow these systems to communicate with each other. Using web services alone is not by far enough to create an SOA; you may e.g. use web

services simply to create point-to-point connections, violating the principle of loose coupling.

19. What is a legacy system?

    Answer:

    A legacy system is a system that the company in question has "inherited", i.e. that has been in place for a long time. In many companies, legacy systems (often large mainframes) run a big part of the day-to-day business.

20. Do you plan to implement SOA using your legacy systems or plan to invest in new tools?

    Answer:

    New tools for implementation, but of course we will be re-using functionality in the legacy systems.

21. If you plan to implement using the legacy system, what are the challenges associated with it?

    Answer: Service-enabling the functionality in the legacy systems.

22. Which one is most efficient, using a legacy system or investing in an entirely new tool (suite)?

    Answer:

    Depends on the scale of implementation and other factors. Most organizations implement SOA not at once, but move through different phases. Implementing a new architecture (or extending an existing one) to an SOA usually implies re-use of existing systems, even if you buy new products to facilitate the implementation of e.g. an ESB or BPM.

23. Which one is most cost effective, building SOA from legacy system or investing in a new tool?

    Answer:

    Same as above, for small implementations it might be more cost-efficient to custom-build, but for larger implementation, using a commercial product will usually be more efficient.

24. What are the techniques you employed to successfully migrate from a legacy system to an SOA?

    Answer:

    - Service-enable the legacy systems using wrappers and, to some extent, writing new code

    - Introduce an ESB and a BPM-tool

- Create and implement a common process in the BPM-tool

- Use the ESB for discovery, routing and mapping (abstraction of the legacy systems)

25. What does it mean to say a system is componentized?

    Answer:

    It means that a system is built up using re-usable components (parts).

26. What are the methods used when converting a legacy system to a service enabled architecture?

    Answer:

    This is not really my area of expertise… But what I know is that there are different ways of accessing the functionality in a legacy system (both when it comes to the technical connectivity and the access to business logic and data). On the technical side, you need some way of accessing the data, this might be a CICS transaction gateway or similar, on top of which you can build a web service wrapper. If your legacy system has native web service capability, you can of course expose functionality directly as web services.

    When it comes to the access to business logic and/or data, it may get more difficult since the legacy system may or may not be built in a fashion that allows you to access the functionality you require. In the best case, the only thing you need to do is to write a web service wrapping some existing functionality. In the worst case, you will have to do quite some restructuring in the legacy system in order to be able to expose the requested functionality as a service.

    Generally speaking, the method you will want to use in order to find out how to use your legacy system in an SOA is to first define the desired functionality based on the business process you would like to implement (top-down approach). You then analyze the capabilities of your legacy system, defining what functionality you can expose and at which cost (man-hours, money, time; bottom-up approach). Most likely you will have to find some compromise between the top-down and the bottom-up service definitions in order to both satisfy the business requirements and not driving up implementation costs too high.

27. Is there any difference between system and architecture?

    Answer:

    Yes. Definition by OASIS: "Software architecture for a system is the structure or structures of the system, which consist of elements and their externally visible properties and the relationships among them." In other words, architecture is like the blueprint for a house, whereas the system is the house itself.

28. How much expertise is available in the organization on SOA and Web Services?

    Answer: SOA: not very much; web services: quite a lot

29. Is this your first time of working on and implementing SOA?

    Answer: *Yes*

    .

30. If No what are the pitfalls of your earlier implementations?

    Answer:

    -

31. What are the lessons learned?

    Answer:

    - *Governance is crucial!*

    - *It takes time to introduce SOA in a company – two to three years at least.*

    - *SOA needs to be introduced step by step; a big bang conversion approach is not feasible.*

    - *Avoid short-cuts.*

32. What was your main goal of implementing an SOA?

    Answer:

    Increased flexibility and reduced cost (in the long run).

33. Do you have external IT consultants working with you when carrying out such a huge project?

    Answer: Yes

34. What is duration of time you consider for implementing SOA 6 months, 1 year or more?

    Answer: More, several years.

35. Can you consider SOA as a solution to IT and business problems?

    Answer:

It CAN be, but it depends on the problems that need to be solved. It's not a silver-bullet solution that fits every problem out there.

36. How do you present the modernization project to the management to secure their financial commitment i.e. financial justification?

    Answer:

    Not been involved in this directly; separate business case for every project using the new architecture. Later projects will have better ROI than earlier ones thanks to re-use etc.

37. Do you have difficulties securing finance from management?

    Answer: No, not initially.

38. Has someone done an ROI analysis of the value of SOA or other approaches for that matter, for the current architecture and reported it to management?

    Answer: Yes

39. Can you change your architecture to accommodate business changes at the speed required by management and the marketplace?

    Answer:

    Hopefully (not in production yet)

40. What do you think is the future of SOA?

    Answer:

    It will become a main-stream architectural approach within a couple of years.