Impact Analysis on Evolution Patterns of Service Oriented Systems

Bureera Sabir  
National University of Sciences & Technology (NUST), H-12, Islamabad, Pakistan

Nabila Perveen  
National University of Sciences & Technology (NUST), H-12, Islamabad, Pakistan

Usman Qamar  
National University of Sciences & Technology (NUST), H-12, Islamabad, Pakistan

Abdul Wahab Muzaffar  
National University of Sciences & Technology (NUST), H-12, Islamabad, Pakistan


ABSTRACT

Service orientation publicized as an important technology for designing, implementing and deploying software systems while promoting coarse granularity and loose coupling. With the changing business needs and requirements developers need to produce software systems using more flexible and dynamic architectures. So that it can be evolved with less degradation risk. Change characterization mechanism in a software allows developers to characterize effects of different changes using various criteria’s, e.g. identification of cause and type of change that need to be made and part of the software where these changes must be implemented. There is a need to identify the impact of evolutionary requirements on system quality. This paper analyzes the evolution patterns of service oriented system and discover impacts which software change characteristics will have on the high-level software architecture. Moreover presents challenges for the researchers in areas of service oriented system maintenance and evolution.

Keywords
Service oriented architecture, Software maintenance, Evolution patterns, QoS.

1. INTRODUCTION

SOA acronym for Service Oriented Architecture, which promotes coarse-granularity and loose-coupling, is the most recent type of distributed software architecture. It composes, classifies and calls components of application in a distributed manner over the Internet. Internet environment dynamics create many challenges for the dynamic service oriented systems. As internet environment becomes more distributive, Service oriented environment also needs to be self-adapted, more distributed and composition emphasizes on better collaboration between services. In addition for software based software’s dynamic and flexible software architecture is required. A strong constraint mechanism is needed to make sure that system runs correctly while describing architectural limitations of runtime systems [1]. SOA based systems give rise to new concerns for software engineers, different from those which were faced in the development and evolution of traditional systems as shown in Table 1 [2].

<table>
<thead>
<tr>
<th>Traditional Systems</th>
<th>Service-Oriented Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known usage patterns</td>
<td>Unknown usage patterns</td>
</tr>
<tr>
<td>Same organization own system components</td>
<td>Multiple organizations own system components</td>
</tr>
</tbody>
</table>

Table 1: Traditional and Service oriented Systems

In service oriented environment the focus shifted from software system implementation to understanding the function of individual components, co-operation and roles of these components within a business process [3, 4].

Roles needed for efficient development, maintenance and evolution of traditional software systems is well analyzed there is an overall agreement on the fundamental requirements and required roles. This latest trend towards the development of systems in accordance with a service-oriented architecture (SOA) methodology has a certain number of repercussions for the above said processes [5]. Nearly all processes involved in the traditional software lifecycle have to be altered to manage service applications. Also, the different tools and roles involved in execution of the processes as well as some organizational policies and structures need to be altered. This paper recognizes the concerns regarding SOA-based systems for the evolution, development and maintenance processes [6, 7]. This paper presents an analysis involving the evolution patterns of service oriented system and identifies the features of a change in software effecting high level software architecture. Moreover presents research challenges in areas of evolution and maintenance of service oriented systems.

2. PROBLEM STATEMENT

The key challenges in software engineering include the design and materialization of intricate software-intensive systems by assembling bits and pieces. Successful component-based software solutions depend on a handy relation between the main business aims, and the flexible IT technical services which permit them. Present software development practices have progressed to create the flexible architectures needed to attain the objectives of component-based approaches like service oriented architectures [8]. SOA abbreviation has now become a well-recognized and divisive term. SOA referred as a loosely-coupled architecture created in order to attain organizational objectives of the business SOA being used to build systems from autonomous services is an architectural approach. Different from traditional systems service integration in SOA environment has become thoughtful. The end system developed in this environment is made up of services coded in different programming languages, is platform independent with diversity in business processes and security models.
Software development process is becoming progressively a universal methodology involving teams from different parts of the world working in unison to attain a common objective. Because of the commercial pressures and the objective of cutting commercial expenses, subcontracting of software development responsibilities from developed countries to developing countries is becoming a common practice in software business. Even medium and small level companies have now opened branches in and many countries. Due to this, software developing teams from all around the world would be involved in the evolution of a single particular product. In light of this, the challenge now is the efficient management and collaboration of the different software development teams spread across the world working together for a common project. Particularly, we could also see such an association of different teams for a single particular phase of the product development cycle. For example, we could have a system designed by a particular team in one location and have it analyzed by some other team in a different location and improvised by a third team again in another location.

During the design phase, the patterns are now a way or a language of communication between different software developing teams for communicating design related issues and solutions [9]. Therefore, design patterns summarize changes in the future which would be affecting only a specific portion of the design pattern. However in general practice the information regarding evolution is not overtly stated in the design pattern document. Due to this fact whenever a change is required, a developer must read between the lines of the design pattern document in order to know the correct possibilities of design alteration. If the pattern is not understood correctly, it might result in different parts missing from the evolutionary process [10]. The constraints imposed by the design pattern must not be violated during alteration of the system parts. Thus, it is important to have information about the pattern design in the documentation of the design pattern. At the design level the evolution of a software system is less costly than it is at the implementation level.

Form years Multi-tiered service-oriented systems have been operational. With current changes in infrastructure technology it is highlighted to understand how Multi-tiered service-oriented systems evolve, what requirements forces the system to evolve and can we derive specialized patterns and trends to be followed by different types of such systems, and to identify in a better way that how these evolutionary requirements effect the overall system quality and actual activities. Recent advancements in legacy applications and their migration to SOA environment is a starting point for this that how this migration effect quality and actual activities. The rest of paper covers SOA in detail and evolvability of service oriented systems.

3. EXPLANATION

Services are the elementary building blocks of SOA. Service is a well-defined program which interacts with other services via a well-defined message exchange. These should be designed for both stability and availability. Services are designed to continue but their aggregations and configurations are designed for change. Software agility being often considered as the foremost benefit of SOA. The idea that an association whose business practices is based over an infrastructure that is loosely coupled is a lot easier to absorb changes as compared to an association which has the constraints of core applications which are a hindrance in the way of smallest of changes. Since the business processes no longer have limitations of the underlying infrastructure, loosely coupled systems will give rise to loosely coupled business processes. Services and interfaces associated with such systems remain more stable and easily accommodate changes to meet changing business requirements. Services in general are used in order to expose the IT investments like legacy and some business applications and these services are then composed into processes, which are then consumed by the user, other systems and services. The composed processes are iterative; they then expose the new services, and compose these services into larger group of the compound applications, which make the output available for the consumption by the users. As shown in figure 1.

![FIGURE 1. BUSINESS-DRIVEN APPROACH TO SOA](image1)

- services,
- service consumers
- SOA infrastructure.

FIGURE 1: COMPONENTS OF SOA

A. Components of Service-Oriented Systems

From an abstract view point service-oriented systems are made up of three major components as shown in Figure 2. The problems and related service orientation issues are categorized in three domains as shown in figure 3.

- Business Domain
  - Engineering Domain
  - Operations domain.

FIGURE 2: Domains of SOA
B. SOA Governance and Life Cycle

SOA governance is the process in which a sequence of responsibilities and communications is created. Control mechanisms, Strategies, and dimensions allows public to accomplish their duties. Service-Oriented Architecture Governance comprises of the following phases [11] as shown in figure 4.

![SOA Life Cycle and Governance](image)

**FIGURE 3**: SOA Life Cycle and Governance

1. Plan
2. Define
3. Enable
4. Measure

The Plan stage describes a governance plan and documents the current IT proficiencies.

The infrastructure and processes of the governance are amends or stated by Define stage.

The Enable stage is responsible for execution of governance rules, infrastructure and mechanisms.

The Measure stage controls the efficiency of the governance and compliance with rules.

Enabling the active processes of the business is promises or key benefit of Service-Oriented Architecture through open, principles-based interoperability. Such principles are of keen importance but we must keep in mind that these principles are not architectures and architectures are not implementations. Good business benefits are not eventually generated by the architecture but the implementation of a architecture which is well designed which results in generation business benefits. Organizations and associations have different expectations from SOA due to the simple fact that different organizations have different business objectives. This is the reason why SOA cannot have a simple enough definition, SOA is a means of getting your objective and it’s not the end goal itself. It is very important to understand the different myths associated with SOA. Table 2 gives overview of the myth or mindset of people about SOA and the actual facts.

<table>
<thead>
<tr>
<th>Mind Set</th>
<th>Reality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service-Oriented Architecture is technology</td>
<td>Service-Oriented Architecture is self-governing plotting knowledge of retailer, product, and technology or some business tendency. Retailer will not always propose a whole stack of Service-Oriented Architecture because SOA requires some variations to another organization from an organization.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service-Oriented Architecture is fresh and innovative</th>
<th>The conceptual examples of SOA were EDI, CORBA and DCOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile entertainment services</td>
<td>B2C</td>
</tr>
<tr>
<td></td>
<td>Mobile Online Games, Music and Videos on demand</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service-Oriented Architecture Reference Architecture diminishes implementation risk</th>
<th>SOA is like a snowflake i.e. no two are the same. It is not necessary that for your organization the best solution is provided by SOA Reference Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>total renovation of processes of business and technology is needed for SOA</td>
<td>SOA is supposed to be depends on recent investments and it should be incremental.</td>
</tr>
<tr>
<td>We are required to construct a Service-Oriented Architecture</td>
<td>SOA is a path way, not an end.</td>
</tr>
</tbody>
</table>

A. Service-Oriented Systems Development Pillars

Service-Oriented Architecture is not just an architectural style, it is a way of promoting applications which needs an alteration of mindset, and which needs arrangement with business processes and goals so that they have some value. The next is collection of best conventions which is termed as the “four pillars”. Figure 5 shows the four pillars of SOA. These pillars don’t ensure fruitful accomplishment of service oriented systems, but many achievement stories proof that the chances of success are limited if these pillars are not acknowledged and admitted [2].

![FIGURE 4 : Pillars of SOA](image)
Strategic Alignment: SOA enables between organizations and systems greater interoperability, quickness, flexibility, leverage of legacy investments, and cost-efficiency. Though, it possibly will result in a costly set of arbitrary services that are never utilized, if select the incorrect approach.

SOA Governance: Ranked as key inhibitor for adoption of Service-Oriented Architecture [12]. Collection of enforcement mechanisms, rules, and policies is provided by Service-Oriented Architecture governance for evolving, developing and using service-oriented systems, and analyzing their business value.

Contextual Technology Evaluation: This pillar requires making sure of the suitability of a technology inside a particular situation, prior to making an enduring promise to it. Some different tools, standards and technologies possibly are elements of an implementation in service-oriented systems.

Change of Mindset: As shown in table 1 the distinctness among traditional and service oriented systems should be dealt with and the myths about SOA should be changed. Processes ought to be targeted at service providers, service consumers or infrastructure providers. It’s better that if these processes are extra associated. Technology assessment becomes an essential element nearly of every life cycle activity, particularly design and architecture.

Every lifecycle activity must include tasks for ensuring the possibility of runtime controlling of service-oriented systems, support for migration towards service-oriented systems development and the changing behavior of such systems.

A. SOA Coding Pattern

In order to accommodate required changes we must understand how Service oriented systems are coded. Figure 6 shows the SOA coding pattern and it comprises of Business Process Execution Language (BPEL), Service Data Object (SDA) and Service Component Architecture (SCA)[13].

Business Process Execution Language (BPEL) is basically a standard executable language used for specifying some activities within business processes correspondence with the web services. By using the web service interfaces the processes in BPEL import and export the information. Service Data Objects (SDO) is designed to simplify the processing of SOA data from varied sources such as Web Services and it can be used with static or dynamic interfaces to make it useful for application programmers and separate the data access code from application code. Service-component architecture (SCA) envisioned for the development of those applications which are based on service-oriented architecture (SOA). SCA includes various programming languages, technologies, frameworks and platforms for Web service components and also for the approaches used to link them.

4. MAINTENANCE AND EVOLUTION OF SERVICE ORIENTED SYSTEMS

A service changes in different perspectives during its lifetime. As a result of this change in perspective a service is most likely to be available in several versions. These changes are

- Interface Difference
- Semantics Difference
- QoS Difference

In distributed service oriented environment peer-to-peer association exists between participants devoid of a central business process. Dynamically constrained evolution should be implemented by the member services themselves. Our aim is to ensure entire system consistency by verifying the constraints on every local service at runtime. Therefore to accomplish this intend each service should save constraints. For distributed service oriented systems service should only consider the evolution or change at its local site, in other words it is unnecessary for a service to check the evolution which has nothing to do with that service.

In traditional Software development process minor changes in business processes are handled in teams whereas most important alterations are handled in projects. To accommodate those changes in business environment following roles are needed [14]

Project Manager (SOA): Is responsible for project management, defining and implementing the Service oriented project plan and monitor the projects

Project Member (SOA): the role of project member may be composed of diverse roles coming from a range of business process teams, SOA quality assurance, Designers and other groups.

SOA directly links business operations to development cycle that’s why it is called system-of-system software which requires strict governance and management. Via configuration management applications relationships management and dependencies of SOA is possible. Systems monitor and record are managed by people. IEEE Software Configuration Management Plan (SCMP) proposed an approach to facilitated governances and control changes in SOA development life cycle utilizing software engineering rules [15].

SOA implementation is a new born concept and researchers are more focused in exploring issues faced in early stages of development life cycle. With increase in SOA adoption there is a need to focus on the impact of evolution on service oriented systems and on their maintenance because of the following reasons

1. service-oriented systems that have been deployed need to be maintained and evolved in order to accommodate changes

2. To make legacy functions available as legacy systems are migrating rapidly to SOA environments

As mentioned in Table 1 service oriented systems differ considerably from traditional systems, As a result of these significant differences many new issues arises which influence sustaining and assessment actions. The distinctions are.
1) Diverse service consumer’s and providers.
2) Short release iterations as a result of changing business needs.
3) Migration to SOA environment with minimal changes in legacy systems.

5. MAINTENANCE PROCESSES
Management of services encompasses many capabilities, these capabilities are

- An ample solution for configuration and change management allowing organizations to provide user with cost effective and quick service updates.
- Complexity reduction linked with Information Technology framework management by focusing on reducing the operations cost.
- Enabling rapid and reliable recovery using centralized backup services and should provide this customer revival without IT interference.
- Pre-deployment capability planned with most appropriate process direction and knowledge explicit to hardware need to assist professional of IT of making architectural decisions at the low-risk.
- Data warehouse and reporting are very important in order to assist IT maintain corporate executive and the quality of services that are provided improved, and properly managing the resources over upgraded exposure capabilities and integration of data management from a wide-ranging of resources.

In SOA environment maintenance processes complexity increases, particularly in a case when there are some external providers and consumers are involved.

- For service providers the impact analysis tasks are possibly unidentified set of users, unless in the SOA infrastructure there are some procedures for tracking consumers’ services.
- For implementation code service the impact analysis the system consumers are considered through interfaces of services and the direct consumer of implementation code service.
- Configuration management starting from what to place in it, business processes, service test instances, services of infrastructure, configuration files, interfaces service etc. becomes more complex.
- Release iterations present among consumers and services, infrastructure and services, and consumers and infrastructure preferably must synchronize, but when these are external then they are not possible.

Shared services between the business processes and consumers make the maintenance more challenging. The processes become a part of SOA governance and needed in order to give the answers of the following questions:

- Who is the responsible for shared service maintenance?
- What will happen when for the same services the multiple business units have different requirements?
- How in the perspective of numerous business processes a service progress?

6. PLATFORM INDEPENDENCE
The major benefit linked with SOA is its platform independence. This benefit is specially associated with web services. This means that underlying services could be implemented in any language. With this massive benefit comes a difficulty especially the recent program analysis tools which code systems in one programming language. From perspective of maintenance and evolution it becomes more difficult where maintainers and developers are not used to with all programming languages which are used for service implementation. Another major problem with platform independence and programmers unfamiliarity is the circumstance of third party services, where only service interface definitions are available. So a detailed documentation of service interfaces should be available from which services behavior can be easily inferred. Elements of documentation should be

- Pre conditions
- Post conditions
- Behavior articulated more formal languages or pseudo code like OCL (Object Constraint Language).

7. CHALLENGES IN MAINTENANCE AND EVOLUTION OF SERVICE-ORIENTED SYSTEMS
Being in crawling phase SOA adoption need more study related to the evolution and maintenance of service oriented systems. As a part of Research Agenda some topics for SOA have been described [16].

A. Dependency and Impact Analysis

According to service provider’s perspective changes in requirements, technologies or business trends lead to service changes. Non adapting service interfaces which do not require modifications have no impact on service consumers. However technology change may negatively affect QoS even if the service interfaces are same. In contrast service interface that require modification imposes a huge impact on service consumers. Extensive Research is needed in the areas related to evolution and maintenance of multiple interfaces, Proper extensibility mechanisms usage in messaging technology (e.g. SOAP), Mechanism of change notification for service consumers, the impact analysis techniques for service providers and consumers.

A rising trend in service oriented systems deployment involves using a set of models that are kept synchronized and need to co-exist. These synchronized set of models can be related to service, access policy, invocations stubs, server deployment description, proxies, service locator interfaces and classes. There is a need of consistent models in such a cases. Research encounters for dependency and impact analysis of service oriented systems include but not limited to the identification of dependencies between models. This will help analyze and better understand, assess and measure the effect of a change.

B. Infrastructures for Change Control and Management

Service-oriented systems are distributed systems and deployed on different servers computers spread over a wide geographical area. Even some of the services used may not be under the control of service oriented system owner. Regardless of the fact that for centralized system configuration management robust techniques are obtainable, there are certain issues related to change management in code repositories and distributed code bases, particularly where there is involvement
of third party system. Moreover for large service-oriented systems configuration management there may be additional requirements as well. An important research challenge is to develop a unified model for change control and management in service oriented systems.

C. Tools, Techniques, and Environments to Support Maintenance Activities

The software reengineering community has established a range of tools and methods to support analysis for maintenance and comprehension of legacy systems. But the development of such methods, analysis tools and specialized methods for maintenance and development of service oriented system is still in crawling phase. Research challenges include but are not limited to find metrics to assess change impact, extraction tools, dynamic and static modeling for extracting information, defined processes for incremental evolution of service oriented systems.

D. Multilanguage System Analysis and Maintenance

Because of the benefit of platform independence an associated benefit is that service oriented systems can be designed, coded and implemented as multi-paradigm systems. Following unique challenges are posed by the maintenance and analysis of such systems.

1) Using dynamic and static analysis tools model, extract and represent information
2) Analyze, explore and visualize extracted information.
3) Process Documentation and support tool for evolution and analysis of multi-language distributed systems.

E. Reengineering Processes for Migration to SOA Environments

Legacy systems or systems migration to SOA environment needs reengineering of that system. The problem with migration of systems to service oriented environment is that recent tools and techniques to appliance portions of SOA migration horsehoe not the full. And important area of research in reengineering process is to develop processes which appliance tools and horsehoe to support other processes. A very complex task would be the automation of such processes. Most of the people working in this area also highlighted that mining of legacy code for the services is the real challenge having high business value. Research areas in this topic moreover include

1) Techniques and tools for the analysis of huge source code to find out part of code that is of value to business
2) To determine reusability of services from business value and metrics for wrap ability.
3) As services respond to features, research could focus on feature extraction applications for identification of services [17]

F. Service-Oriented Systems Evolution Patterns

From years many researchers focus on the evolvability of the systems. Such studies and papers help maintenance organizations to accomplish tasks in a better way by identification of portions of systems which are stable and those that are prone to changes. Research areas could be identifying evolution aspects with specific focus on service oriented systems. And to identifying metrics which determines services resilience to change.

8. CONCLUSION

Service-Oriented Architecture is not just an architectural style, it is a way of promoting applications which needs an alteration of mindset, and which needs arrangement with business processes and goals so that they have some value. Organizations these days are rapidly moving to SOA environment to make use of benefits such as agile business processes, cost efficiency and adaptive systems with leverage of legacy systems. Benefits of SOA adoption are realistic but not automatic. Organizations need to align business strategy and SOA strategy, establish SOA governance processes, and perform contextual technology evaluation of technologies of choice for their SOA implementation to realize these benefits. Organizations must have to accept that to implement service oriented systems there is a need to change mindsets because SOA adoption requires changes which will be reflected throughout the service oriented system life cycle.

This paper includes a detailed study of service oriented systems with a specific focus on their maintenance and evolution. We then identify the key implications and research challenges in these areas. Then highlighted the use of IEEE standards based SCM which makes use of software engineering practices and rules to manage complex changes in service oriented systems and challenges faced by implementers which implement changes to such systems. Using accurate identification, control, evaluation / review of configuration items and characteristics we can release changes in environment in an efficient way.

9. REFERENCES


66


[10] Jing Dong, Sheng Yang, Dushyant S. Lad, Yongtao Sun “Service Oriented Evolutions and Analyses of Design Patterns”[2009]


