

Table of contents

1	Introduction	15
1.1	Motivation	15
1.2	Problem Statement	16
1.3	Existing Approaches	17
1.4	Limitations of Existing Approaches	18
1.5	Goals	18
1.6	Proposition	19
1.7	Contributions	21
1.8	Organization	22
2	State of the Art Work	24
2.1	Background	24
2.2	Related work on SOA	29
2.3	Challenges of SOA	34
2.4	Cloud Computing	35
2.4.1	Cloud Computing Providers	39
2.4.2	Software as a Service in Cloud	40
2.4.3	Related Work to SaaS	41
2.5	Related work on Security Access Controls	42

2.6 Openaource Security	45
2.7 Business Process Reengineering (BPR)	48
2.8 Background of BPR	48
2.9 Factors of BPR	49
2.9.1 Success Factors	49
2.9.2 Failure Factors	51
2.9.3 Benefits of BPR	53
2.10 Business System Reengineering	53
2.10.1 The Goals of System Reengineering	53
2.10.2 System Reengineering Approaches	55
2.11 Using Data Flow to Explain System Reengineering: A Case Study	57
2.12 Summary	60
3 ASOA in Cloud Computing	61
3.1 Development Process Methodologies and ASOA	61
3.2 An ASOA Designed Service at Work	73
3.3 ASOA Solution's Progress Monitoring	75
3.4 ASOA and Critical Factors	77
3.5 ASOA for Cloud Computing	79
3.6 Summary	80
4 Cloud Computing Secured Gateway	81
4.1 Cloud Server Side Gateway Algorithm	81
4.1.1. Database Notation (SQL comparison)	82
4.1.2. Notations	82

4.2 Mathematical Description: Cloud Server	83
4.3 Client Service Linear Analysis	89
4.4 Mathematical Description with Linear Analysis: Cloud Client	90
4.5 Cloud Secured Gateway	96
4.6 Summary	97
5 Cloud Security Gateway – Use of ASOA	98
5.1 ASOA proposed Service Framework	98
5.1.1 Use of Map-Reduce in ASOA	99
5.1.2 Use of Parallelism	102
5.2 Use of Service in the Cloud	103
5.3 ASOA Use Cases	106
5.4 Summary	117
6 Master Observer Service (MOS)	118
6.1 Master Observer Service	118
6.2 Challenges associated with Services' Performance	120
6.2.1 Availability of proper bandwidth	120
6.2.2 SLA issues among involved service providers	120
6.2.3 Surge in Cloud access	121
6.3 Service Governance	121

6.4 Service Security	124
6.5 SOA 3.0 MSS Model	126
6.6 Analysis and Results	128
6.6.1 Services Failure Prediction	128
6.6.2 Success of Services Utilization	134
6.7 Closing Remarks	134
7 Conclusion and Future Works	136
7.1 Conclusion	136
7.2 Realization	137
7.3 Limitations	139
7.4 Future Work	141
8 References	144
Addendum	149

Tables

TABLE 2.1: A PRICE COMPARISON OF DATA CENTRE SERVERS	37
TABLE 3.1: SRD NOTATIONS USED IN THESIS	62
TABLE 3.2: SEQUENCE OF ACTIONS OF A BUYER	74
TABLE 3.3: ADVANTAGES OF AN ASOA SOLUTION TO AN ORGANIZATION	78
TABLE 6.1: ATTACK ON THE SERVICES	132

Charts

CHART 1: THE EXPECTED FAILURE OF NEXT MONTH	130
CHART 2: ATTACKS DEPICTION ON MEAN IS 0.1	132
CHART 3: ATTACKS DEPICTION ON MEAN IS 0.7	133
CHART 4: ATTACKS DEPICTION ON MEAN IS 1.0	133
CHART 5: ATTACKS DEPICTION ON MEAN IS 5.0	133

Figures

FIGURE 2.1: SERVICES	25
FIGURE 2.2: A DISTRIBUTED VIEW OF SERVICES	25
FIGURE 2.3: THE USE OF SERVICE-ORIENTED ARCHITECTURE	26
FIGURE 2.4: BANK CUSTOMER SERVICE SOLUTION	27
FIGURE 2.5: A HORIZONTAL FRAMEWORK OF SERVICES	30
FIGURE 2.6: FLAT SERVICE FRAMEWORK	32
FIGURE 2.7: STRUCTURE OF TREE BASED FRAMEWORK	32
FIGURE 2.8: A CLOUD WITH SEVERAL DATA CENTERS	36
FIGURE 2.9: A DATACENTRE	36
FIGURE 2.10: A COMBINATION OF SERVICES FOR CONSUMERS	38
FIGURE 2.11: CLOUD COMPUTING	39
FIGURE 2.12: BUSINESS PROCESS FACTORS	50
FIGURE 2.13: AN OVERVIEW OF SYSTEM REENGINEERING	54
FIGURE 2.14: LEGACY SYSTEM	57
FIGURE 2.15: REENGINEERED CLOUD SYSTEM	59
FIGURE 3.1: A COMBINATION OF SEVERAL SERVICES	63
FIGURE 3.2: A FUNCTIONAL EXAMPLE OF AN SRD	63
FIGURE 3.3: A WORKING SERVICE PROTOTYPE	65
FIGURE 3.4: ANOTHER PICTORIAL VIEW OF THE SRD	65
FIGURE 3.5: A GENERAL ASOA SOLUTION MODEL	70
FIGURE 3.6: A DRILL DOWN TO PROCESSING PHASES	71
FIGURE 3.7: A DETAILED PROCESSES VIEW IN ONE CENTRAL PROCESS	71

FIGURE 3.8: A SERVICE BLOCK	72
FIGURE 3.9: A CONSUMER’S ORDER TO BUY A PRODUCT	73
FIGURE 3.10: THE COMPLETION OF PURCHASING AND SHIPMENT PROCESS	74
FIGURE 3.11: A COMPLETE ASOA DESIGN OF SECURED SERVICES	76
FIGURE 3.12: A COMBINATION OF SERVICES FOR CONSUMERS	80
FIGURE 4.1: ASOA UE IN SECURED GATEWAY	81
FIGURE 4.2: CLOUD SECURITY GATEWAY	96
FIGURE 5.1: COMBINATION OF SEVERAL SERVICES IN A CLOUD	99
FIGURE 5.2: MAPREDUCE IN ASOA PROPOSAL FOR CLOUD	100
FIGURE 5.3: THE USE OF SERVICES USING MAP-REDUCE	102
FIGURE 5.4: SERVICE DISCOVERY PATHS IN A CLOUD	103
FIGURE 5.5: REQUESTORS SERVICING INPUT	103
FIGURE 5.6: REQUESTORS SERVICING OUTPUT	104
FIGURE 5.7: NOTATING SERVICE M AS A START/END SERVICE MARKER	104
FIGURE 5.8: SERVICES AVAILABLE IN THE CLOUD	105
FIGURE 5.9: UTILIZATION OF TWO DIFFERENT SERVICES S1 AND S2	105
FIGURE 5.10 LOGIN – USE CASE A	107
FIGURE 5.11 LOGIN – USE CASE B	108
FIGURE 5.12 LOGIN – USE CASE C	109
FIGURE 5.13 LOGIN – USE CASE D	110
FIGURE 5.14 LOGIN – USE CASE E	111
FIGURE 5.15 SESSION CHANGE – USE CASE A	112
FIGURE 5.16 SESSION CHANGE – USE CASE B	113

FIGURE 5.17 SESSION CHANGE – USE CASE C	114
FIGURE 5.18 PAGE REQUEST – USE CASE A	115
FIGURE 5.19 PAGE REQUEST – USE CASE B	116
FIGURE 5.20 LOGOUT – USE CASE A	117
FIGURE 6.1: SERVICES AVAILABLE IN THE CLOUD	122
FIGURE 6.2: AN SOA 3.0 MODEL OF INVOLVED STAKEHOLDERS	126

Abbreviations

ASOA	Achievable Service Oriented Architecture
BPR	Business Process Reengineering
CSM	Cloud Security Model
CSP	Cloud Security Protocol
CTO	Chief Technology Office
DFA	Demand For Alteration
EUD	End User Development
ISP	Internet Service Provider
JAD	Joint Application Development
MIS	Management Information Systems
MOS	Master Observer Service
MSS	Messaging Security Service
OS	Operating System
P2P	Peer to Peer
PC	Personal Computer
QoS	Quality of Service
RBAC	Role Based Access Control
SaaS	Software as a Service
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOA-DP	Service Oriented Architecture Documentation Process
SOAT	Service Oriented Architecture Team

1 Introduction

The advent of Cloud computing has introduced a way to add capabilities dynamically by increasing the capacity of new infrastructure using service oriented architecture (SOA) without newer investments. These investments can include the issuing of new licensing, the training of users, and the provisioning of external user friendly interfaces. Cloud helps organizations functional and data processing in increased volume output with fewer people working on the organizational software applications. The reduction of capital spending on technology infrastructure is another key benefit. The ease of access to information for any given entity is now available with minimal upfront spending as the model of “Pay as You Go” or is based on-demand. Another benefit is the globalization of the workforce to expand the business with less cost, by streamlining the processes using both closed and/or Opensource technologies.

1.1 Motivation

The Cloud has brought new extensions in the world of social and corporate sectors with available resources of available Information Technology provided by both Closed Source and Opensource Technology Solution providers with the data availability in the shape of Big Data within the Cloud. Organizations need to use Business Process Reengineering (BPR) to adapt to either Closed and/or Opensource solutions. In the past few decades, we have seen growth in Cloud computing in the IT industry and Closed source technology providers did capture several

organizations to serve, however in last decade or so, organizations are starting to move to Opensource technology utilization.

1.2 Problem Statement

The merger of Cloud and Big Data in the past few years after the emergence of Cloud computing has introduced gaps in terms of standard Business Processes as well as Data Security in the areas of Role Based Access Control (RBAC), and other data security protocols within the realm of the Opensource world of technology. These gaps are the problem area that we have explored to provide an Opensource secured protocol for any organization or entity to tailor according to their environmental constraints.

The gaps can be understood in terms of the example of Opensource technologies and Closed source technologies within an organizational entity. Both technologies utilize different sets of protocols. Opensource protocols are available to alter or enhance according to the needs of stakeholders, however closed source technology providers usually do not allow any such alterations, and rather they customize their products.

The use of Service Oriented Architecture (SOA) is yet to be explored in all of the aforementioned, as once Unified Modeling Language (UML), Business Execution Processing Language (BEPL) and standard System Development Life Cycle (SDLC), such as Waterfall or Spiral models, were utilized in pragmatic application/software design and development. The users of Big Data and Cloud need to go through Business Process Reengineering (BPR) using secured access to their work sessions to process their data using MapReduce, which is mostly security

agnostic in an inter-organizational environment. The question of how we can use BPR in such a way that is helpful to make the right choices to start utilizing Opensource technology is addressed. There is another gap that is researched in this thesis and that is the lack of Opensource solutions of standardized observer services, as are provided by the Closed source providers to monitor secured message deliveries to and from or within an organizational Intranet or VPNs.

Another question that is concentrated on in this thesis is how to use SOA in a way to extend and leverage its usefulness, as well as its enhanced embodiment of solution design in such, which allows architects and software designers ease of use for creating a visual presentation using one specific methodology rather using several such as BEPL, UML, and/or CMMI, etc.

1.3 Existing Approaches

The contemporary industry has been using structured, object-oriented, and web services, for quite a while to program software applications. However, Cloud has introduced a paradigm shift, where applications are written in terms of Cloud Services, to perform processes or to be used for message passing within or without applications used by organizations for transaction management, order processing, and secured sessions, while using Unified Modeling (UML), Business Execution Processing Language (BEPL), and Rational Unified Processing (RUP), etc. Organizations have also been utilizing Pipe and Filter, Client Server, Three Tiered, and Service Oriented Architectural styles to create software applications and design systems to deal with the complexities of organizational processes to serve both internal, and external end-users.

1.4 Limitations of Existing Approaches

Cloud Computing has broadened the software applications domain for both individual and organizational uses. The approaches that software developers have become accustomed to, such as structured, object-oriented, and RESTful web services, needed some extension, which could have been utilized as a gap filler. We have seen the emergence of Service Oriented Architectural or SOA styles to be adapted to design software architectures to create software applications and design systems. To further extend SOA, this thesis elaborates the extension in terms of Achievable Service Oriented Architecture or ASOA.

1.5 Goals

It is one of the prime attractions for any organizational sector to reduce capital costs on their technology projects. There's no need to spend big money on hardware, software, or licensing fees. Software engineering organizations must improve accessibility by having access to our software applications anytime, anywhere, making life so much easier. Cloud is used majorly these days, and there are Closed source security solutions available by major organizations. There has been a need of an Opensource security protocol that organizations can customize according to their needs. The research presented in this thesis is to provide such an Opensource solution for the industry.

1.6 Proposition

This PhD thesis proposes an extended use of Service Oriented Architecture (SOA) for creating a Cloud Security Protocol (CSP) by providing a security model in terms of Business Process Reengineering (BPR). The proposed solutions presented in this thesis elaborate on use cases embodied within a department of Computer Science of a university. These do not discuss the required functionalities and capabilities in traditional available web security models used within contemporary business environments.

This thesis introduces a model, which is called Cloud Security Protocol (CSP), which is based on Achievable Service Oriented Architecture or ASOA. ASOA is an extended use of SOA. This thesis concentrates down the use of MapReduce [1]. This concentration introduces Business Process Reengineering (BPR) that can be layered as per the security aspect encompassing the data and related processes within the elements of transparency as applied to cloud service providers to service consumers, who are involved in any type of analytics and would like to explore Opensource solutions.

The use of the Business Process Reengineering (BPR) approach is given in detail in Chapter 3, the actual implementation in terms of contribution to this thesis, where we will explore the extended use of SOA is provided in Chapter 4, and this BPR targets at improving security features over traditional existing models in terms of Closed source technology providers, while not taking any risks by threatening other important features of the current security models.

This thesis also presents the Secured Gateway Algorithms using CSP in Chapter 4 to reduce the threat of data loss or prevention of any outside invasion to the security of an organizational data asset(s) within the Cloud. The primary purpose of the Cloud Security Protocol (CSP) and the

elements of transparency are to generate evidence-based confidence between users and the services. All users, who are involved within a certain analytical processing in their sessions within the cloud, in terms of security, need to work in an environment, which assures secured access to their data as well as applications. It does not matter if these applications of data are related to a closed system or associated to Big Data Analytics. The linear analysis is also given of presented algorithms to show the time complexity.

A Master Observer Service (MOS) is also introduced in this thesis as an extended contribution within the Opensource realm and is customizable according to an organizational/entity's needs. MOS using a Messaging Secured Service (MSS) within a Private Cloud is provided in Chapter 6, and refers to the stipulation of different services required by users' on-demand in the form of computational resources as a combination of services. The services are designed and developed using Achievable Service Oriented Architecture (ASOA). Any software application that requires frequent modification needs to be separated from the servicing applications, which are consistent and rarely need to be updated.

In a nutshell, this thesis presents four major contributions towards the Opensource community to their parameters within their environment to work securely, while transitioning to Big Data adaptations for any given needs. To solidify this research work, there has been several publications published in IEEE, ACM, Elsevier Science Direct and Springer conferences and research journals.

1.7 Contributions

This PhD thesis sheds light on the following in terms of its contributions:

- **Use of Business Process Reengineering (BPR)**

The work produced in this thesis is built upon the foundation of BPR, since the algorithms, protocols, and processes are now well defined with the technology utilization, we have maintained the idea of Business Process Reengineering to extend Service Oriented Architecture.

- **Achievable Service Oriented Architecture (ASOA)**

Use of SOA has been done in terms of creating procedures to write web and RESTful services, which can interact with users and each other, in terms of functional as well as data processing alongside of message passing needs. ASOA has been provided in this thesis as an extension, with a service design methodology to create Cloud Services.

- **Cloud Security Protocol (CSP)**

CSP is an Opensource Security Protocol that any organization can adopt and customize accordingly to provide their internal as well as external users and related stakeholders.

- **Master Observer Service (MOS)**

MOS is an adaptation of CSP to prove that CSP is customizable using Opensource technology. Any organization can adopt and customize accordingly to monitor the use of service utilizations.

These are the proposed artifacts in terms of Opensource methodology enhancements provided in this thesis, with which Cloud users will get access to the secured session(s) to work within, which will be layered as per the security aspect encompassing the data and related processes within the elements of transparency as applied to Cloud service providers to service consumers while using Opensource technology, e.g., Amazon/Microsoft Azure, etc.

1.8 Organization

This thesis is organized in a way to shed light on the relevant background in detail to build upon software engineering methodologies to Cloud computing techniques using a novel methodology ASOA [2], which provides the modeling for the proposed security protocol (CSP) for secured session management. This thesis discusses in the form of chapters, where Chapter 2 is the State of the Art, and the rest of the chapters provide the contribution by the thesis author and the advisors:

Chapter 1: Introduction

This chapter describes the problem of lack of Opensource security within Cloud. It also presents the contributions of the author as well as the structure of the thesis.

Chapter 2: State of the Art Work

This chapter shed light on the background state of the art work conducted by several researchers on the areas of SOA, Business Process Reengineering, Opensource Security and Quality of Service.

Chapter 3: Business Process Reengineering (BPR)

This chapter provides the details on BPR, the adaptation of BPR and the paradigm shift within contemporary business with practical examples.

Chapter 4: Use of Achievable SOA (ASOA)

This chapter presents the extended use of SOA with detailed discussions on the use of SOA with key notations and practical examples.

Chapter 5: Cloud Computing Secured Gateway Algorithm with Linear Analysis

This chapter presents detailed algorithms with linear analysis for the Opensource adaptation by any organization according to their contextual needs.

Chapter 6: Master Observer Service

This chapter discusses a concept of a monitoring service with secured message passing for any processing needs of an organization. The mentioned concepts are also implemented on a real time scenario and empirical results are presented in both tabular as well as charts forms.

Chapter 7: Conclusion and Future Works

This chapter concludes the research work and contributions, as well as the future work associated with the propositions presented in this thesis.

2 State of the Art Work

Service Oriented Architecture (SOA) is an approach that aims for adaptability and flexibility, it is not a technology. SOA facilitates the utilization of reusable IT components in terms of software applications, to create new solutions over an existing framework of components. It provides platform-independent coupling of service components. This coupling can involve diversely developed service components in various programming languages, which can be maintained on several operating systems. The logical or functional separation of service components is provided by SOA [1]. This separation allows software designers and developers to modify, test and/or redevelop, and run these components on different servers before initiating them into a new lineup.

2.1 Background

Let us take an example to explore SOA within an academic research continuum. Both request and response, as given in Figure 2.3, can be considered as a service provided by a retailer, travel agency, bank or a government service provider, etc. These services are available around the world, as an offshoot of global business. Figure 2.2 depicts the servers providing these distributed services across the globe [2]. Using the Cloud, these services can be accessed almost anywhere in

the world. The utilization of SOA works for consumers as well as industrial internal-operational and managerial users for an organization.



Figure 2.1: Services [2]

SOA has evolved in stages over the last few decades, since industrial computerization increased. The services we use today process requests as input and produce output for customers, other systems or services. These systems or services orchestrate the data when generating messages among each other and to the user. The operational user of these services can be monitoring or managing many requests simultaneously. A distributed view of services given in Figure 2.2 shows the Cloud around the globe.



Figure 2.2: A distributed view of services [2]

These operations can also be performed by a mediator service designed to follow the agreed upon policies and procedures among each of these services. One of the major contributions presented in this thesis, is precisely providing an Opensource Cloud Security Protocol (CSP) in Chapter 5, which can be customized according to the organizational internal security policies and practices. Each service is owned and governed by a business entity and works within a certain body of rules, defined by the policymakers.



Figure 2.3: The use of Service-Oriented Architecture

Standard SOA provides a service data abstraction. This abstraction can be understood as a services' messaging metadata, which can be in the shape of XML, XSD, JSON or other set industry standards providing interoperability to a user's request. The messaging between services is encapsulated for information-hiding purposes. An actual SOA solution for a bank as shown in Figure 2.4 is designed to maintain customer service [2] as well as web self-service options for consumers [2].

SOA eases the development of ever-changing applications, which can compare data with stationary applications while maintaining a decoupled relationship between these applications on a simultaneous basis as well as maintaining Quality of Service (QoS). ASOA [3] is also known as Achievable SOA, and it is discussed later in this thesis in Chapter 4, by introducing a fresh approach for business information technology, making it easy to assemble and configure IT components like building blocks that can be combined to provide easy and fast creation of

solutions. For example, a bank provides a line of credit by checking a consumer's credit; an automotive parts seller checks the inventory; a shipping company maintains the shipping status for delivery to consumers, etc.

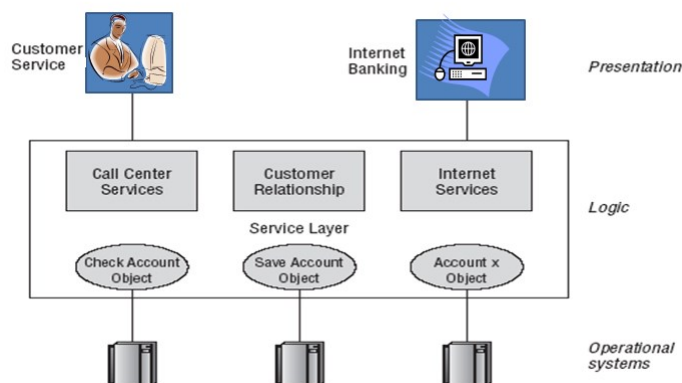


Figure 2.4: Bank customer service solution.

According to Schreiner and Lamb [2], systems of the future will be based on the concepts of SOA. Service applications will be composed of a number of individual services running at several servers. As illustrated by Erl [4], service component application logic can be divided into two levels: a service interface, where loosely coupled services are available with their implementation and technology platform; and a service using application in which service application logic is developed and deployed on different technology platforms.

The purchase planning scenario [1] of a commercial or residential property is an example of a system where the property-selling agency can be provided a flexible approach using SOA. The levels discussed above can be adapted as application and service interface levels. We now take a closer look at how SOA helps to plan a purchase of a property and adapt the “composed property purchase service.” Initially, three services [1] can be identified as:

- i) “Property for sale” search service
- ii) Offer submission service to initiate a purchase
- iii) Credit check and mortgage service

In this example, partner service implementations are selected as follows. The “property for sale” service can be a local server provided by the real estate agency. The purchase offer submission service can be the web services provided by another agency, or it can be assumed by the law firm. Such legal purchase services may provide the possibility of applying web service policies, but apart from that, the services might not be configurable. The credit check service is a web service provided by the financial department of a bank or a mortgage provider. Considering the strict security requirements of a financial transaction, these services typically have fixed functional and non-functional requirements—i.e., security policies—that cannot be altered.

By using SOA, these services can be combined at one front-end platform on a website to provide the “property for sale” information and other services as mentioned above. To make this clear, we can take the example of Amazon’s [5] store-front, where customers use a browser to get the displays on Amazon.ca. The front-end website gathers the signals or the customer’s intent triggers and calls upon services that do things like acquiring the data for the current on-sale products, or getting the customer’s order. The important thing to note here is that the servicing components do not make proposals to the customers, and these services have no idea who they are talking to. These services are serving customers by getting data from some other services, which might be residing at some other server and can provide only product details. Some other services might be obtaining customer order details to invoke other services for shipping the products.

The business sector is most of the time searching for ways to convert its Information Technology or Management Information System departments to swiftly adapt to changing business needs by either adopting closed or Opensource solutions.

2.2 Related works on SOA

SOA is an approach that provides the methodology to form web services as functional building blocks to build an infrastructure for consumers, which is not platform-dependent. Platform independence makes it easier to develop web services, such as operating systems, programming languages like Scala and Akka provide functional aspects, and Java is famous for Object-Oriented aspects of software development. Quality of Service (QoS) is an important factor in SOA as well. We live in an information age, managed by electronic devices. These devices communicate with each other using data communication networks, the best example here being Cloud. QoS for networks is an industry-wide set of standards and mechanisms for ensuring high-quality performance for critical applications. By using QoS mechanisms, network administrators can use existing resources efficiently and ensure the required level of service without reactively expanding or over-provisioning their networks

Several researchers, Korostelev et. al., [7] Borcoci1 et. al., [8] Zeng et. al., [9] have focused on the domain of web services, while working on QoS-aware service composition and management. The composition of QoS-aware services in Peer to Peer (P2P) networks has engaged much of the researchers' interest. For automatic maintenance of QoS levels research in a P2P infrastructure, Huynh et. al. [10] have proposed self-organized and QoS-aware Virtual Service Communities. Gu et. al. [11] examined large-scale P2P systems in detail for their QoS-aware service compositions. They introduced SpiderNet, which is capable of doing:

- Service path selection
- Service path instantiation
- Benefit-driven peer clustering

This P2P framework model produced results of service provision. Let us take an example of an insurance company, which sells the following business services:

- Home insurance
- Health insurance
- Mortgage insurance
- Employment insurance
- Critical illness insurance
- Automotive insurance

We can see these services as the following traditional business framework.

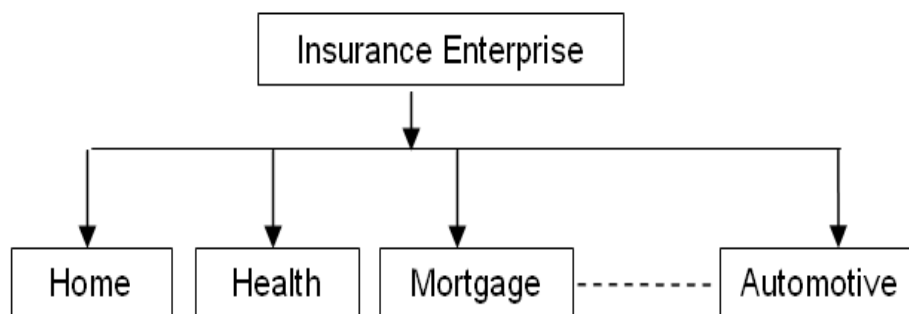


Figure 2.5: A Horizontal Framework of Services

Let us explore this horizontal framework of business services. All of the mentioned business services are directly related to one enterprise in the mentioned insurance company. When a consumer needs one insurance product, the insurance enterprise must appraise all of the service provisions and locate the appropriate service(s) from the listed services. The analysis for the needed and related services can divide the service hierarchy into several choices, and then sends these choices to the consumer's requested service(s).

From the researchers' point of view, a structure for service management is required to maintain QoS in a SOA solution. The aim of this research [12] [13] is to construct a framework that supports QoS for a SOA solution. SOA permits business organizations to interrelate their information systems using predefined rules for mutual dealings within the organization and with consumers. SOA is a methodology that interconnects businesses and computational assets, such as software in use, current system applications and human resources of the organizations involved in the design, development, and deployment of a SOA solution. These assets are used to achieve results with the use of newly deployed service(s) for consumer(s). These consumers can be the customers as well as the operational users. The use of web services is increasing to address the need for more consumerism and enhancement of business across the globe.

The QoS-aware service [10] [11] [14] management and composition are garnering attention from academia as well as being established as industry research topics with the progress of SOA development. Challenges are arising for QoS-aware service management and composition with the increasing use of distributed P2P networks and the use of several technologies.

The conducted experiment provides a comparison between QoS-aware hierarchical service composition and management system design with commonly used flat-based service composition and management as shown in Figure 2.6, and it was found to outperform against the flat-based

service(s). The results showed a higher success rate for satisfying the user's QoS requirements. Xie *et al.* [14] proposed a design of a QoS-aware hierarchical service composition and management system as shown in Figure 2.7 in a context of a JXTA-enabled P2P network.

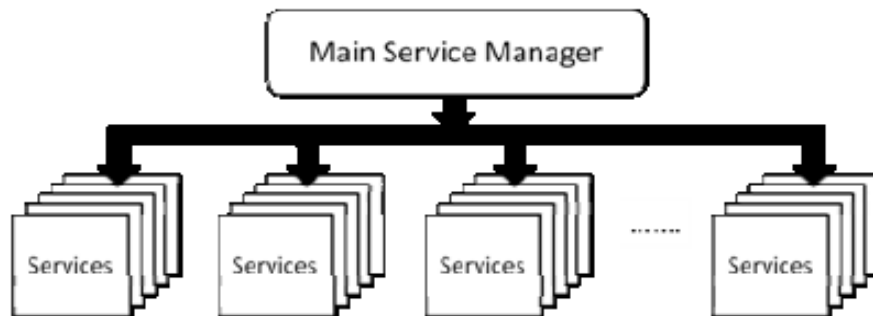


Figure 2.6: Flat service framework [14]

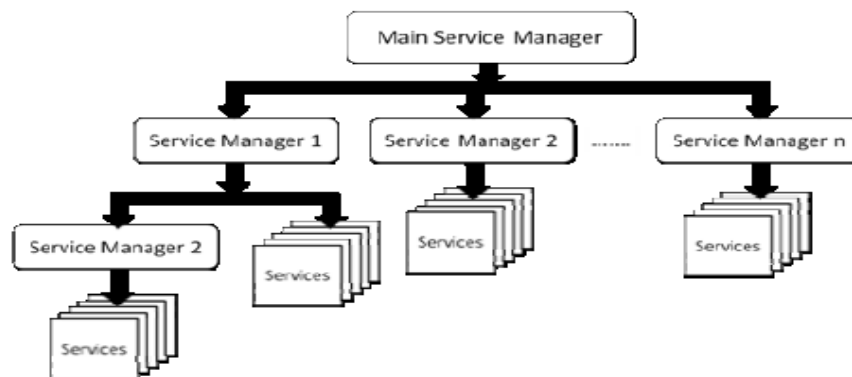


Figure 2.7: Structure of Tree based framework [14]

Research conducted by Xie *et al.* [14] focused on the design and the implementation of a tree-based hierarchical QoS-aware service composition and management scheme in a JXTA-enabled P2P environment. They examined ways of an efficient service management system, which can assist with and can be improved for user applications' QoS requirements. The verification of this improvement was found by implementing the hierarchical service composition and management system design in a real JXTA-enabled local network.

Software architecture is a vital part of software engineering, and SOA presents an architectural concept. Dorner *et al.* [15] have brought forward a few considerations of SOA in terms of End User Development (EUD). They analyzed the development of adaptable systems as a potential for SOA, proposing challenges that need to be solved to get an effective EUD. The authors' analysis is based on requirements for EUD systems and empirical studies, taken from earlier research work [16]. Dorner *et al.* have suggested in their study, that SOA can be extended with structures for in-use modifications; the design of user-adaptable next-generation systems is also possible.

With SOA-provided flexibility, the new customizable systems can be produced, and platform independence can also be achieved. Services designed using SOA are formulated software applications, and this formulation is closer to business domains. Research presented in this chapter has also indicated that the call for additional metadata of service descriptions is growing quickly. The amount of data collected from use experiences of a service, needs to be stored for analysis of service and its future use. This is also known as sentiment analytics. This data handling, in terms of storage locations and synchronization, raises issues and serious concerns about service performance. The service can have performance issues in terms of message communication to and from the user, and to the service provider, due to this additional contextual information. This research [15] presented has found that requirements of EUD of a service may involve extending protocol and server structures of SOA standards.

2.3 Challenges of SOA

This pursuit presents some challenges as given below [17, 18, 19, 20, 21]:

- Flexibility is required.
- Increasing demand of standardized services with seamless experience for consumers.
- Reduction in operational cost of these services by getting satisfactory results due to improved efficiency, which means users control their business, rather than technology.
- Services are mostly distributed.
- Getting regulatory approvals.
- Getting rigid information systems wrappers developed to get the services combined at one framework, such as COBOL & RPG application transformation might need Java program to create a wrapper of data extraction for Big Data Analytical system.
- Efficient use of existing resources.
- Services are heterogeneous.
- The patterns of services interaction are unpredictable
- The service(s) “front end” is less useful for testing purposes due to the decoupled implementation on the backend.

SOA characterizes and provides the composite framework for the Cloud services based on open standards. Multiple services can be combined to serve a consumer at one independent Cloud; for example, a travel agent can provide car rental, hotel reservations or can book an entire vacation. This is a convenient way for the consumer to get several services with one phone call or one website utilization. American Express travel [6] can be taken here as an example.

The services designed with transparency to the background technology and business processing management rules, for all internal and external users, offer the highest ongoing returns in terms of feedback for the industry's investment of skilled personnel's time. The utilization of Opensource technology tools segregates service application's front-end and back-end programmers from the specifics of any single platform and operating systems to produce transparent, portable, and coded pieces of an application, which can later be combined as needed.

2.4 Cloud Computing

The Cloud is a loose union of millions of computers all over the world, which provides data and available "Software as a Service (SaaS)" to process, display, and communicate data between users in order to share and exchange data in textual, voice or video formats. The Cloud offers millions of users a way to find and share their relative information, conduct research, and accelerate learning, etc.

The servers that compose the Cloud are situated across the globe. A user's need to process any data over the Cloud is served by any or many of these servers, using software applications designed and developed for the Cloud in the form of services. Several of these services are freely available for use, such as the GoogleTM search facility or AmazonTM's book search utility. The servers providing these services are stationed at some datacentres. The cloud is a combination of all of these datacentres.



Figure 2.8: A Cloud with several Datacentres

The datacentres [22] are a home for servers. Software utilized to run these servers is provided by several organizations at a cost. This cost involves licensing fees, upgrade patches development cost to the software providers, etc. The hardware itself comprises several components. Figure 2.9 shows a portion of a datacentre containing servers in racks connected to each other in a central network and linked with routing servers to serve the clients.

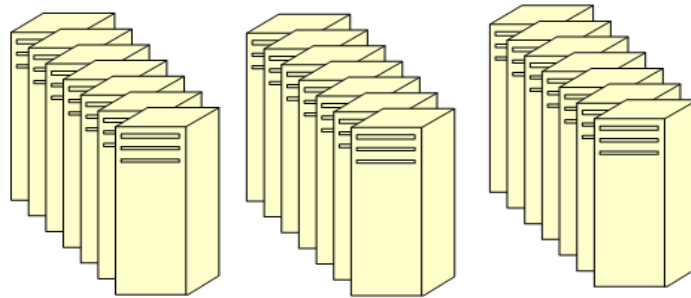


Figure 2.9: A Datacentre

Running a datacentre to provide services to users in a cloud has an associated cost. A discussion of this cost is given below to understand why the Cloud will be introducing costs to a user in the future. The cost of one small business server to address intra-organizational needs in a rack is given in Table 2.1.

IBM [23]	DELL [24]	HP [25]	MenDax [26]
US\$ 1,649.00	US\$ 1,168.00	US\$ 1,104.00	US\$ 1,109.85

Table 2.1: A price comparison of Datacentre Servers

To run a datacentre, a complete infrastructure is required; this infrastructure contains a building with large rooms for racks that store servers, with network routers connecting these servers together and with outside organizations providing data communication services, electrical power to keep these servers running, a controlled temperature environment and much more.

We can see the datacentre layer 1 in Figure 2.10 contains data processing and storage servers, connected with each other for efficient and fast data transmission. Layer 2 contains smart routers to balance the incoming request load(s) [27] for layer 1 processing, with firewalls to provide communication security. Layer 3 contains request receivers and response broadcasters.

Computers, and thin clients such as Blackberries, iPhones, laptops, etc., use the Cloud to communicate in synchronous and asynchronous ways to one another as loosely coupled networked devices. There are several forms of such connections found in the Cloud. Some computing devices are directly connected to others with a wire or fiber-optic cables, and some are connected through local modems. Thin clients use satellite communications.

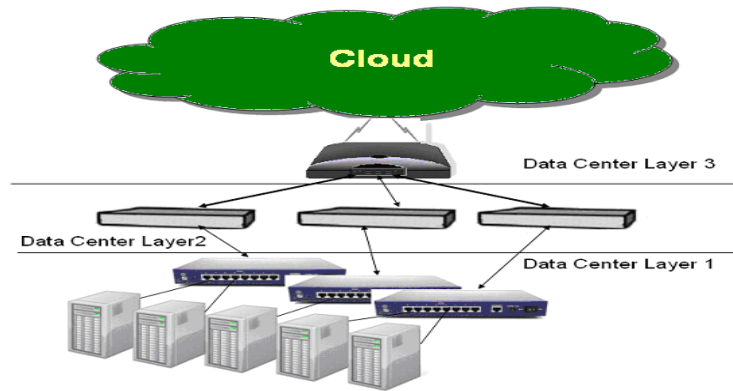


Figure 2.10: A combination of services for consumers

Basically, the Cloud is a combination of services for everyday users, from several service providers. Members of the business community use the Cloud for its transparent and platform independent features to assist each other in providing services to their consumers. As this thesis presents work on Opensource security protocol for the Cloud, according to the definition by Gartner [28], “*Cloud computing is a style of computing where massively scalable IT-related capabilities are provided “as a service” using Internet technologies to multiple external customers.*” Figure 2.11 depicts a Cloud Computing scenario. We can consider the following as services:

- Software
- Local Device Operating Systems
- Cloud-provided Operating Systems
- Web Browsers
- Networks and related protocols, etc.

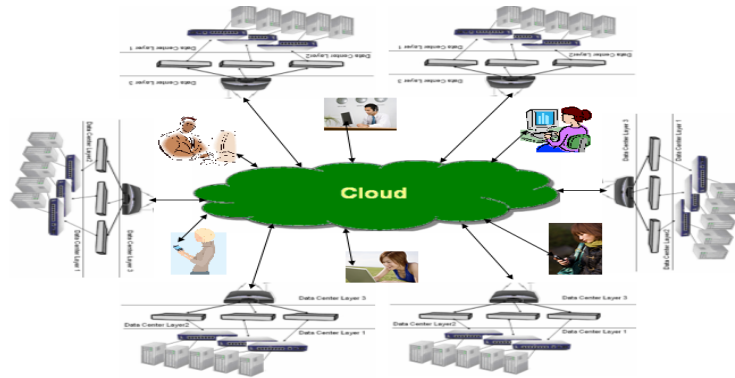


Figure 2.11: Cloud Computing

2.4.1 Cloud Computing Providers: Cloud computing is currently being offered by several service providers. We will look into a few of these as an example, such as Amazon Elastic Cloud [29], EMC Atmos [30], Aptana [31] and GoGrid [32].

- Amazon Elastic Cloud [29] (EC2) is a web service, along with Amazon SimpleDB, Simple Storage Service (Amazon S3), CloudFront and Simple Queue Service (Amazon SQS), which provide resizable computing capability with web-scalable computing for developers.
- EMC Atmos [30] provides information distribution and multi-petabyte storage. This service merges automated data placement with substantial scalability to efficiently deliver content and information services anywhere in the world. EMC Atmos operates as a single entity, using metadata and business policy on an automatic basis to get the right information to the right location, at the right time. These combined features increase a user organization's operational efficiency, reducing its technical management complexity, and is a cost-effective solution.

- Aptana [31] Cloud provides on-demand scalability, pre-configured applications, Cloud management tools, and access to hosted source control and staging servers, so users can work more efficiently for the sake of their business and their customers.
- GoGrid [32] provides an interface for consumers by offering a multi-server control panel. This interface allows consumers to organize and administer load-balanced Cloud servers in the period of a few minutes.

2.4.2 Software as a Service (SaaS) in Cloud: Software as a Service (SaaS) is a model whereby software is provided by a vendor, such as when an ISP or a cable company provides an Internet connection and charges its services for a certain period, in a monthly or quarterly fee for the provided service(s). The service provider in this case is the IT infrastructures of a hosting company, which runs the software at its datacentre and lets the consumer use its services under a certain agreement among service provider and the user. Software as a Service (SaaS) is an on-demand commodity and is available in Cloud from several service providers. SAPTM and OracleTM are good examples of SaaS providers. The users pay only for the services they use.

Disadvantages of SaaS can include limited personalization or customizing of applications as the core functionality is outsourced and the user of an organization, which is utilizing SaaS has to alter procedures according to available modules of the application(s) being used by one's organization. If the service provider's network has a malfunction, the user(s) of the mentioned organization cannot take any immediate steps to route business procedures to another server or another service provider.

SaaS is advantageous to smaller and medium-sized organizations, as SaaS can be adopted with ease of installation. This saves organizations excessive costs for technical staffing. The sales

and marketing force of the organization can work from home or on the street, in a shopping mall, etc. Management can even work while traveling, meeting with clients or addressing other official business. Organizations save by not installing expansive hardware as mentioned in Table A. Much of the network in-office cabling is not needed as well. Google [33] has started providing Google Drive; this service is free and allows the user to process several types of documents, with the use of a simple browser, employing several rich features like those we utilize in our daily routines in document-processing software on our local computing device.

2.4.3 Related Works to SaaS: As per Goth [34], Software-as-a-Service (SaaS) has acquired the focus of software and related technology people on the new business model that enables software on-demand for an organization. Goth also illustrated in his article that pioneers of SaaS have seen [34] *“a new era dawning where the obstacles to communication between users and programmers—and sometimes the lines differentiating those roles themselves—are much diminished.”* The rise of SOA design and advancement of new development models, as well as Web service standards, will cause change in solutions’ development in a way that deviates greatly from traditional software development modeling methodologies.

The usage growth of SaaS in several organizations is over 20% a year [35], in comparison to single-digit traditional software growth. Sathyan *et al.* [36] have introduced a method to facilitate customers, in search of services across multiple service providers, to obtain the benefits of services offered by other service providers simultaneously. The authors also have taken the basic tenet of unified service experience into consideration in the design of this new approach. They claim that their approach is standards-based and maintains the utilization of existing messaging protocols to transfer inter-services messaging. Sathyan *et al.* [36] ensured their approach would have lots of flexibility with regards to the client, in terms of OS, platform, etc.

They also considered the convenience of the customer's service experience, while concealing from the customer the inherent complexity involved in providing the service assembly. The use of SaaS, SOA solutions, and Cloud computing is the upcoming paradigm shift from traditional legacy computing. This shift does have software as the main element.

2.5 Related work on Security Access Controls

It is vital to have our enterprise secured. Security issues are a serious measure of implementation within any enterprise. In the contemporary industrial environment security requirements are approached to consider the need for models that gather the organizational and distributed aspects of information usage within and without an enterprise. Access control is a crucial part to be looked into, before we deploy any software application. A security model provides a strict depiction of the access control security policy and its employed method(s). The reinforcement allows the proof of properties on the safe keeping for data critical to the organization provided by the access control system being designed [73]. The scenario, where authorization-to-user is the most significant entity and access control decisions are repeatedly determined by employee functions in an enterprise level software access both within and/or using a VPN.

Access control of an application, when used from Cloud focuses on an end-user's functions and access right(s). This end-user function, also called a role, which we consider to allow an application use. There are several roles an end-user can have within an enterprise, such as a manager, administrator, accountant, office staff, internal customer, etc. Since the inception of distributed computing, we have seen many current access control models such as the Chinese wall

model, Role-based Access Control (RBAC), Task-based Access Control (TBAC), Context-based security model, etc. We encapsulate the state-of-the-art for these access control models.

The Chinese Wall model [74] aims to preclude delicate information concerning an organization from being revealed to competitor organizations through the work of financial consultants. The Chinese Wall Model vigorously institutes the access rights of an end-user based on what the user has already accessed [74]. However, the stringent implementation of the access rules to use applications results in their being inelastic and limiting.

The permission to access data are assigned to end-users using Role-based access control (RBAC) [75]. This practice within an organization simplifies the user management and assists to determine efficiently whether the availability of certain information access permissions are to be granted for certain users. However, the environment of user roles of end-users is static, which means RBAC lacks flexibility and responsiveness, as well as does not encompass the overall context associated with any collaborative activity [76]; RBAC can be considered a passive security system that assists the function of maintaining user permission assignments. Furthermore, the lacking of the capability to stipulate a fine-grained security mechanism on individual users in certain roles and on individual object instances [76], which means there is not enough for collaborative environments, which can dynamically have secured access using RBAC.

Task-based Authorization Control [77] (TBAC), is a task-oriented and an active model. This model allows for management on dynamic basis of permissions as tasks progress to completion [76]. The permissions are related with contextual information in TBAC, which means it is about ongoing activities when evaluating an access request from someone for any processing needs, these authorizations have a stringent runtime usage, with legitimacy condition, and

expiration characteristics [76]. TBAC only keeps track of usage and legitimacy of permissions, however, there are no contexts covered in relation to activities, tasks, or workflow progress. This lack of mentioned properties is insufficient for collective software applications, which involves a much more extensive definition of a context both within and without an enterprise.

The Context-based security model [76] uses contextual graphs. This model provides an appropriate technique for stipulating security requirements in prevalent environments for organizational security needs. By using this model, we can create a security management application that can provide the ease of security administration for involved stakeholders for complex environments with many heterogeneous services and devices, such as the use of VPN, Cloud, and Web-based software solutions. The study [76] shows the inclusion of contextual information, policy specifications, and policy enforcement characteristics, as well as fine-grained control, and groups of users. However, this model is based on distributed environments, and requires further testing within the Cloud based computing [78].

Since, the inception of Cloud computing and later Big Data Analytics environments, we have seen the growth of organizations in a large-scale collaborative environment in terms of systems and do have many cooperative organizations as a substitute of competitors. So the Chinese Wall model is not suitable for any system that deals with Big Data within Cloud. In a standard SaaS, authorization focuses on an end-user accessing the service, it does not deal directly with end-user's task, nor the context, which is being processed for any Data Definition or Data Manipulation as in standard RDBMS in terms of DDL and DML, which also caters Analytical processing. The task-based authorization control and context-based security model are not suitable for Big Data Analytical needs either. In TBAC, the access control decisions are often determined

by the end-user roles and control of safeguarded resources, in terms of anonymous analytical processing to produce trends for business decisions.

Big Data Analytics use-cases are easily divided into groups by Predictive Analytics, trends generation using Statistical techniques. The function of an end-user is a defined role. There are levels of access certain users get and few get permission to produce analytics. Cloud Security Protocol (CSP) presented in this research work assigns permissions to an end-user to reduce the complexity of security administration. Chapter 5 sheds light on this aspect with several use-cases in pictorial representations.

This chapter has discussed the state of the art research work that has been done by several scholars, and the next section starts to build on Business Process Reengineering (BPR), of available application design methodology, which is UML to transition into the proposed ASOA service design methodology with details in upcoming chapters.

2.6 Opensource Security

Since the emergence of Big Data the security has been an issue. A project for security Big Data in Cloud is called the Apache Knox Gateway. This is a REST API [79] [80] Gateway to work together with Hadoop clusters to manage Big Data in secured way for any processing or ETL reasons. The Knox Gateway is an Opensource initiative to provide a single access point for all REST processing with Big Data and Hadoop clusters.

It is able to deliver valuable functionality to support in the governance, incorporation, observing and mechanization of precarious organisational and systematic needs of an organization.

- **Authentication:** The providers, such as Hortonworks, Cloudera, MapR or IBM provide with the role of authentication to the Big Data Cluster to allow end-users to perform ETL. They are accountable for gathering credentials accessible by the REST API consumer. They validate the credentials and transfer the end-user with pass or fail result for the ETL processing. Apache Knox provides support for:
 - OpenLDAP,
 - ApacheDS and
 - Microsoft Active Directory.
- **Federation:** Any organization (Public/Private), when requires credentials to be accessible to a restricted set of confidential entities within a closed domain, the federation is achieved by getting the Knox Gateway configured for authenticating identity of an end-user for an external authentication request to process any desired ETL processing.
- **Authorization:** The authorization for any ETL processing requires of an end-user or application's role for the requested resources from a Hadoop cluster managing Big Data needs to be based on the effective user identity context. The rules need to be mapped by the Knox provider to determine the identity context.
- **Audit:** It is vital that an audit trail is available for any legal or compliance check reasons. This trail provides an ability to regulate the actions for keeping the ETL safe and the actions performed over Hadoop cluster can be revisited.

The Knox Gateway also balances the Kerberos protected Big Data cluster having coupled with suitable network segregation of a Kerberos protected Hadoop management for Big Data. The protection of Big Data policy enforcement is done by Knox API Gateway using a reverse proxy and the backend services for which proxies are requested. These policy enforcement deals with:

- Authentication
- Federation,
- Authorization,
- Audit,
- Dispatch,
- Host Mapping and
- Content rewrite rules.

Supported Hadoop Services: The Knox Gateway is integrated with the following:

- HDFS – Hadoop Distributed File System
- HCatalog
- HBase – Hadoop Database
- Oozie
- Hive/JDBC
- Yarn Resource Manager
- Storm

Apache Knox delivers a pattern driven technique of adding new transmitting services. This supports for different Hadoop REST APIs to originate quickly and easily, since it uses XML for message passing. It also permits users and developers to supplement provision for custom REST APIs.

2.7 Business Process Reengineering (BPR)

The business environment is continuously changing at a very fast pace as Cloud computing is taking over as a new business environment to assist organizations. To cope with the issues of process changes, such as adaptation of new or different rules and regulations at the time of an expansion of a business extends a need of its business processes to be reengineered. This section discusses in detail about three fundamental concepts of Business Process Reengineering (BPR), Systems Reengineering, and SOA. This section also provides background of these concepts as well as provides the artifacts researched and work done by several researchers and business analysts. We will look into the proposition of extending the use of SOA in terms of ASOA.

2.8 Background of BPR

The term business process reengineering (BPR) was coined by Michael Hammer and James Champy in 1990 [37]. The two pioneers define business process reengineering as “*the fundamental rethinking and radical redesign of business process to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service, and speed*” [37]. However, achieving dramatic improvements through business process reengineering is easier said than done. According to Abdolvand *et al.*, [38] 60-80% of BPR have been unsuccessful. The high failure rate coupled with huge costs make BPR a very risky operation.

After the brief description of system reengineering and service-oriented architecture (SOA), we will discuss each of the common factors that could make BPR a success and how they apply across

cultures. We will also discuss the common causes that make BPR a failure and how to prevent them from happening.

System Reengineering can be described as focusing on improving existing or legacy systems for the new need or reducing operation cost. In some articles [43, 44], Reengineering is closely related to maintenance, which is defined by the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment. By placing Reengineering on a continuum with maintenance and new development, the true nature of Reengineering becomes apparent. Maintenance entails making corrective, perfective, and adaptive changes to software, while development focuses on implementing new capabilities, adding functionality, or making substantial improvements typically by using new computer resources and new software technologies.

2.9 Factors of BPR

This section presents a detailed discussion on the concepts worked by several researchers.

2.9.1 Success Factors: Abdolvand *et al.* [38] came up with five success factors, which they called positive readiness indicators as shown in Figure 2.12, and they are: egalitarian leadership, collaborative working environment, top management commitment, supportive management, and the use of information technology. The authors based their research on two Iranian companies in different industries as well as research done by other experts. Each main factor has sub-factors. The combination of the factors and their sub-factors cover all the important factors [38].

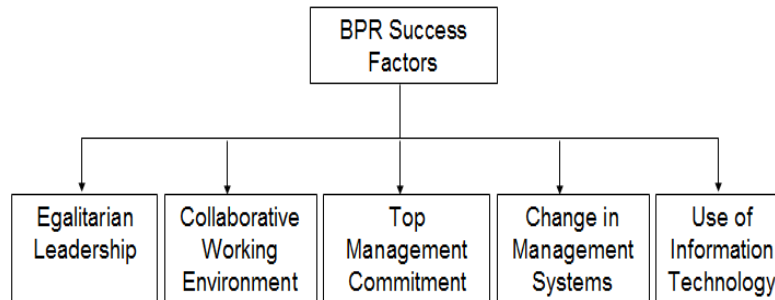


Figure 2.12. Business Process Factors [38]

A critical success factor is what Abdolvand et al, [38] called egalitarian leadership. Top management should drive the changes by providing vision and employees should become more responsive. However, while communication is very important in the success of any venture, nonetheless, culture plays an important role in how people communicate. For example, in low power distance cultures such as the United States and some West European cultures, relations between leaders and the average person is much closer than other cultures that have high power distance. Such cultures (high power distance cultures) always expect ideas to come from the top in the form of commands and the rest are expected to obey without questioning the wisdom of such ideas. Would this egalitarian leadership approach work universally? We believe it is highly unlikely to work across cultures.

The second critical factor for success is collaborative environment where employees need to work together in a friendly way to achieve a common goal [38]. This approach has a better chance of success if applied universally. Even the more individualistic cultures in the West believe in team work. Meanwhile, this approach could also be applied in collectivistic cultures (most collectivistic cultures happen to be high power distance cultures). As a result, collaboration stands a good chance in working universally across cultures and, hence, makes BPR successful.

The third and fourth success factor is the commitment of top management with supportive management of an organization. Management should have a clear strategic vision, or, in other words, they must know where the organization is and where it is headed. The authors [38] were short on description on this topic. However, we believe without managerial commitment and involvement, the organization could easily fail in realizing its objectives. Management needs to guide the path and offer support to their employees in order to transition seamlessly into their new environment. For example, one way management could support its employees is to provide training for their future development.

The fifth success factor is the use of information technology. As per Abdolvand *et al.*, [38] “*IT is introduced as a critical component and even a natural partner of BPR, which has a continuous and important role in BPR projects*”. In fact, IT is a natural “ally” of BPR. It is intrinsically an integral part of the process. However, organizations need to make due diligence before undertaking the proposed change. They must calculate their return on investment (ROI) before investing on technology infrastructure.

2.9.2 Failure Factors: Abdolvand *et al.* [38-44] cited resistance to change as the most common factor that causes the failure of BPR. The authors called this factor the “unreadiness factor”. Humans resist change, particularly, when they don’t expect to benefit from the change. We believe since humans are self-interested, and some employees may even go to great lengths to derail the process, management must reassure its employees that they won’t be worse off when the process gets completed. The fear factor is a legitimate concern which needs to be addressed before the change starts.

Another factor that could contribute BPR to fail is having unreasonable expectations about the project. According to Herzog *et al.* [41] “*The first consists of training about the importance and role of BPR, about the benefits of BPR, the role of cooperation, the commitment to employee training, and the resource availability for training*”. Herzog *et al.* [41] argue that management must be knowledgeable and have realistic expectations about the benefits of BPR and what it can actually contribute to the bottom-line.

One often overlooked factor in BPR failure is culture. We already mentioned some success factors that may not work across all cultures. According to Newman *et al.*, [41] “*The tenets of Confucianism suggest that a BPR effort will challenge Chinese ideology and the workforce will resist radical change including changes to the management hierarchy. Confucius considered it important to maintain a state of harmonious equilibrium in society*”. The authors found that Chinese companies considered cultural issues as the most important factor when undertaking BPR. Newman *et al.*, [41] mention that Chinese culture prefers gradual change, and BPR advocates a dramatic change; hence, most companies would not adopt BPR in China.

Moreover, due to the risk-averse nature of Chinese society, Newman *et al.*, [41] also have found that “*most Chinese managers advocate stability and want to maintain power and control. Indeed, many employees are satisfied with getting explicit work instructions and are not willing to initiate actions that are often associated with risks*”. We agree with the authors. Usually, uncertainty avoidance cultures, which include Chinese culture, try to avoid taking chances. Such cultures do not like ambiguities. Since BPR is fraught with uncertainty, it may not work in those cultures.

2.9.3 Benefit of BPR: Organizations need to make a thorough research before they decide on implementing BPR. As it is mentioned earlier, BPR is extremely expensive and has a high failure rate. Therefore, the organization(s) must collect all the information needed before any decision is made. Moreover, organization(s) must establish metrics that help analyze the cost/benefit of a new project [41].

Having metrics in place ensures that an organization knows what it is getting into. However, in order for reliable metrics to be established, management must be knowledgeable [41]. There is no black and white answer to this question and experts disagree on this. Some suggest if an organization is doing fine in terms of business processes, it should not rush into BPR; making changes gradually might be enough. However, prominent experts on the field such as Michael Hammer [37] advocates no less a change than a complete (100%), fundamental, and a radical change. Hammer says don't do changes piecemeal [37].

We suggest that BPR may not be the best approach for some organizations. The factors that could influence an organization's decision on whether to implement BPR or not, could include evaluating where the organization stands versus the competition.

2.10 Business System Reengineering

For quite a while in the last four to five decades systems were being created for industrial requirements, design decisions processes and business rules. In order to effectively use current assets of an organization, it is important to develop a systematic strategy for the continued evolution of currently legacy systems to meet changing mission(s), technology and users' needs.

Moreover, knowledge of the business rules and technical decisions is often embedded in the code of conduct in organizational policy and software programs. Such knowledge is difficult to recover after many years of operation, evolution, and personnel change. The software portion of a legacy system may have been written 10 or 20 years ago, developed using archaic and ad-hoc methodologies, and subjected to prolonged costly maintenance. The result is a legacy system that lacks the ability to evolve and develop for meeting changing demands. Systems reengineering is the requirement to fill this gap. Years ago, legacy systems were created for customer and business environment needs and used by companies.

2.10.1 The Goals of System Reengineering: The given figure 2.13 shows the goals of system reengineering:

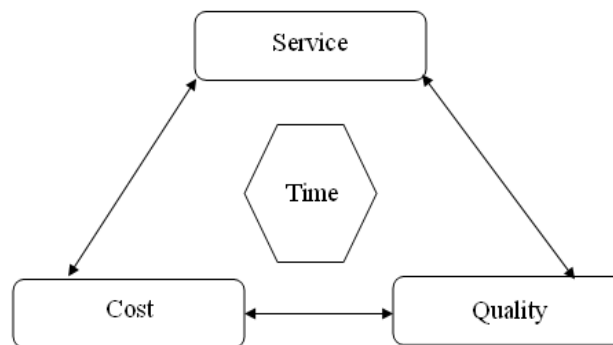


Figure 2.13: An overview of system reengineering [42]

Reengineering is the transformation in a systematic way of an existing system into a new form in order to improve an operation, system capability, new function and high performance at a lower cost, and less time with high quality service to the customers. The word system is a combination of business rules, processes, and operations of user applications. Software is usually used to elaborate a system in the world of Information Systems.

2.10.2 System Reengineering Approaches: There are three main approaches [42] utilized to reengineer any system, the goals of each approach are as follows. The first one is "Big Bang" approach, and this replaces the entire system at one time. The main advantage of this approach is that it can be used by projects that need to solve an immediate problem, such as migration to different technology altogether, or totally different user interface as well as the system architecture. The disadvantage is the result tends to be on the bigger software projects or Enterprise Resource Planning (ERP) software, such as SAP, may not always be suitable. For large systems, this approach may consume too many resources or require time before the target system is produced. The risk with this approach is high as the system must be functionally intact and work in parallel with the legacy system to assure functionality.

The second approach is the "Incremental" approach to a software system re-engineering. It is also known as the "Phase-out" approach. In this approach, system fragments are re-engineered and added incrementally as new versions of the system are needed to satisfy new goals that contain the training of staff or an individual, to learn operating error free module one by one. The project is broken into re-engineering fragments based on the existing system's fragments. Every module of the software should have similar functionality with some enhancements with the new technology, which is developed and tested on the basis of the legacy software module(s). The main advantage of using this approach is having less risk than the "Big Bang" approach as the risks can be identified and monitored at the time of development and Beta testing of each module. The disadvantage of using this approach is the use of a much longer timeframe to reach to the goal of reengineered software.

In the "Evolutionary" approach, as in the Incremental approach, fragments of the original system are replaced with newly reengineered fragments. The components of the legacy system

need to be broken by use and are reengineered into new modules. The main advantage is the converting of a legacy system into the latest technology. There is one disadvantage and it is that similar functions must be identified throughout the legacy system then refined as a single functional module.

There are quite a few factors involved in the “migration” process of a legacy system to be converted into a WEB user interface (UI) system, such as the coding language, and the base technology that contains RDBMS, operating system and the new networking technology, which involves wireless networking as well as advanced hardware infrastructure. System reengineering activities in general require techniques and tools that help to dig out valuable knowledge about the structures and inner mechanism of software systems from the source code using reverse engineering, since other information sources such as system documentation and developer’s knowledge are often not available or consistent.

A key dilemma during reverse engineering tasks is that the information extracted from the source code has to be condensed and abstracted towards an abstraction level which can support software engineers to perform essential analyses and/or modifications of the system, while preventing them from an information overflow that makes software reengineering goals complicated. These activities have to be supported by tools. An important consideration in building tool support for reengineering is what information must be provided, and how this information is modeled.

2.11 Using Data Flow to Explain System Reengineering: A Case Study

Let us assume that there is a branch of a retail company found in Grand Forks, ND USA. The legacy system of this company is that: as a first step, a customer picks up goods in the branch of the company. After that, the customer pays the bill and is serviced by a clerk at the front desk. Moreover, the clerk hands in transaction information to another clerk who works in the reporting department. Once this process is complete, the process moves to accounting of the goods, reports containing transaction information are sent to the headquarters in Grand Forks. Once the headquarters receives the information from the branches, it will send the required goods to the branch. Figure 2.14 shows the data flow of the business process in Unified Modeling Language (UML).

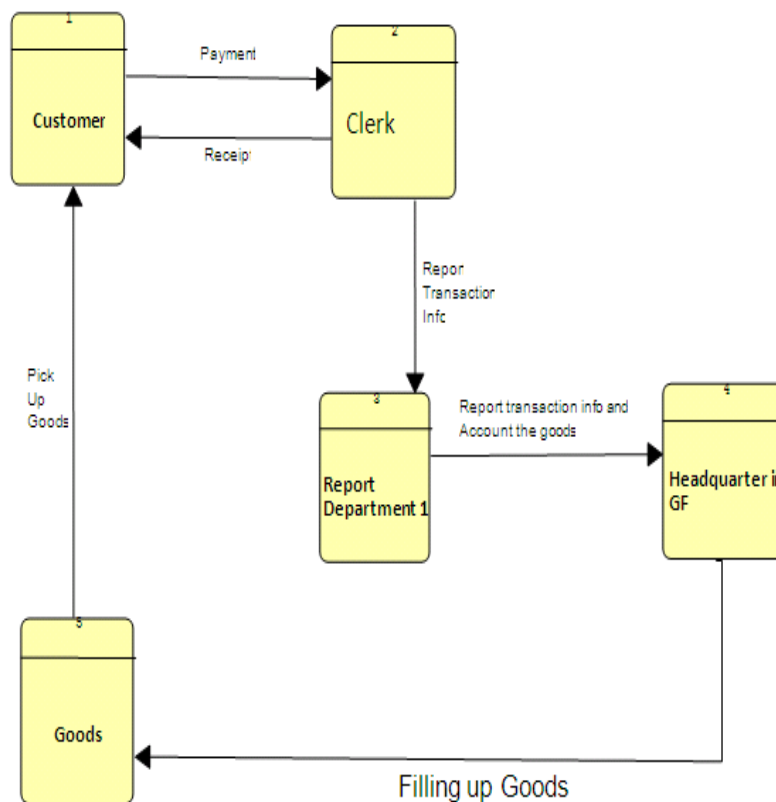


Figure 2.14: Legacy system [45]

Let us have a look at a reengineered system for the provided legacy transaction processing system currently adapted by the company in Grand Forks. This reengineering introduces the retail company system with the given benefits as described:

- The service time is to be reduced, as the clerk would enter requirements of goods at a Cloud UI, once the internal customer's purchase of the goods for initiating restocking purpose at Grand Forks branch.
- The cost of the goods could be reduced significantly per branch, since the Cloud system is reengineered using Opensource technology and there are no licensing fees for the third party software provider(s).
- System reengineering introduces an improved business operations cycle, by connecting all branches with the headquarters via Cloud.

There are a few problems addressed below, which will help us understand the system's fragments:

- a) Specifications of a model to represent design information of a mini or main-frame system.
- b) Specify operations to query the design information stored in the model.
- c) Specify operations to abstract low level program entities into more high-level entities using Entity Relationship Diagrams (ERD) and Data Flow Diagrams (DFD).

Once this task is over the system reengineering can be successful. The current wave of the Cloud computing utilization of Service-Oriented Architecture (SOA) needs both reverse as well as system reengineering. An organization can take advantage by developing web services to be

deployed in the Cloud for the expansion of their processes beyond their internal infrastructure to its consumers and other outsourced partners.

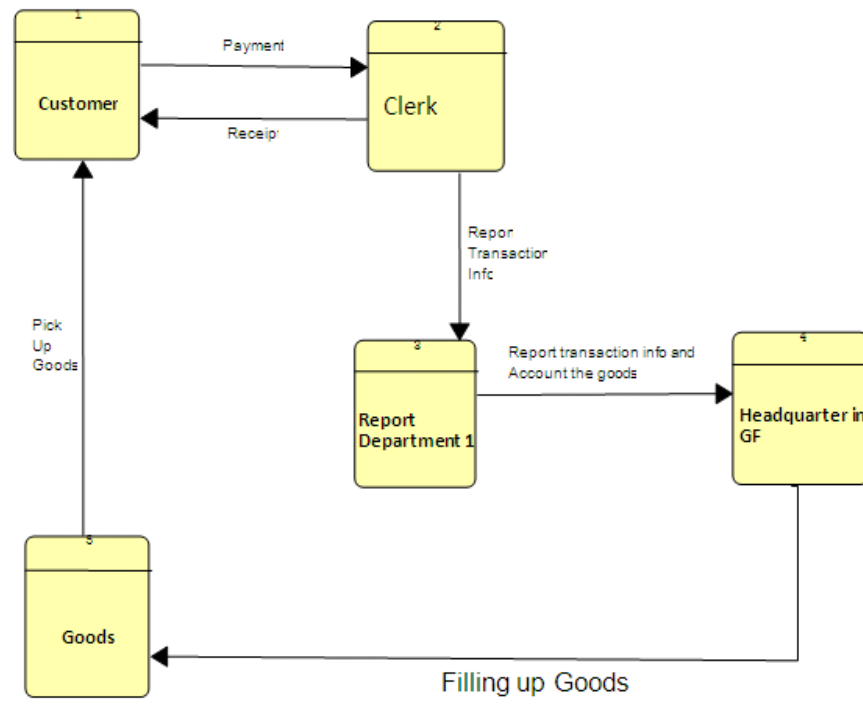


Figure 2.15: Reengineered Cloud System [45]

Business Process Reengineering (BPR) is a relatively new paradigm. Since its inception in 1990, numerous organizations have tried to adopt BPR. However, the high failure rate of BPR indicates that it is not beneficial for everyone, as mentioned previously in this chapter. Moreover, the cost of failure is very high in terms of financial loss and morale of the employees affected. Before an organization tries to adopt BPR, they must make sure of their standing against their competitors. If they are way behind, then they have no choice but to adopt BPR.

In addition, since culture plays an important role in determining the success or failure of BPR, organizations must consider the cultural norms before making a move. A knowledgeable management can help set realistic goals and expectations that will reduce employee anxiety about

the changes to come through effective communication, in case the BPR is the only choice. We would like to conclude that the risks outweigh the benefits, and most organizations must not try to adopt BPR unless they feel they have nothing to lose.

Software architecture is a vital part of software engineering, and the use of Service-Oriented Architectures (SOA) for Cloud deployable services presents an advanced architectural concept with significance for a system's transition from a legacy state to a Cloud state. With SOA-provided flexibility, the new customizable systems can be produced, and platform independence can also be simultaneously achieved. Services designed using SOA are formulated software applications, and is closer to business domains. Research has also indicated that the call for additional metadata of service descriptions is growing quickly, and the amount of data collected from use experiences with a service needs to be stored for analysis of service and its future use. This data handling in terms of storage locations and synchronization raises issues and serious concerns about service performance. The service can have performance issues in terms of message passing to and from the user to the service provider due to the add-on additional contextual information.

2.12 Summary

In the contemporary world of business around the globe, BPR combines both systems reengineering as well as SOA to be a real catalyst of change towards the prosperity of an organization willing to adapt and change with the pace of time. This chapter has detailed the state of the art work, which has been published in relation to Software Engineering, Business Process Reengineering (BPR), and Information Security. Let us have a detailed look at ASOA in Chapter 3, with a few use cases, which are built upon ASOA methodology.

3. ASOA in Cloud Computing

This thesis work has contributed to the Opensource community and the work includes both academic as well as Opensource research conducted within the contemporary industry as well. Almost all of the organizations do data gathering activities [46], which identifies the artifacts and the relationships of a system. It gathers data from system examination, document scanning, and experience captured. As per the contemporary Big Data and NoSQL world, it is noticed that the logic can remain the same as it was in legacy systems, however the message passing, or app to app communication, is the key to manage data for analytic reasons and needs intelligent novel designs of algorithms.

3.1 Development Process Methodologies and ASOA

ASOA [21] is a relatively newer introduction with Achievable SOA adaptation with the use of System Requirement Design (ASOA-SRD) methodology embedded within. It is an approach to design a SOA solution to represent, revise and develop software systems or services required by any organization as a stakeholder for Cloud deployments. It incorporates the elements of requirements into a finished software system in a deployable combination of applications with a single frame of reference. The fundamentals of the SRD are requirements of requesting stakeholders distributed in elements, once amalgamated in the phase of processing to produce a block of an application for deployment.

A block in SRD is a combination of several elements. The design of an application or, in view of SRD, a block starts with an “Initiation Marker” as given in Table 3.1. A design combining several blocks can be shown by this marker only, there can be several blocks (applications) involved in a finalized system. Requirements contain raw material elements in a scattered form. Software engineers use their skills to combine these elements in a processing phase to build an application block. The building of these blocks to achieve targets can also be understood as the required performance of the deployed application.




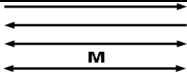
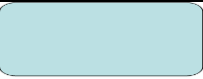

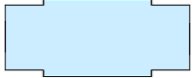

Task	Notations
Marker	
Requirements Elements	
Single Element	
Communication Links	
Processing Operator	
Conditional Operator	
Block (An Application)	
User	

Table 3.1: SRD Notations Used in Thesis

Concerns about the behaviour of different elements of requirements and how they might differ from each other is addressed using the communication/messaging links, which establish a relationship in a logical order among these elements. These communication links specialize into one to one, one to many and many to one, or many to many element linking.

A communication link can be labeled with a textual tag by the software engineer to communicate a consequential relation connecting the two linked elements. These links are ubiquitous and provide the system designer with a constructive element(s) relationship representation for non-technical stakeholders.

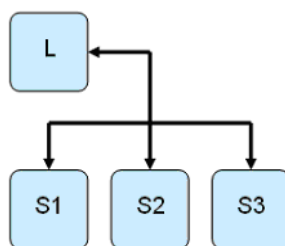


Figure 3.1: A combination of several services

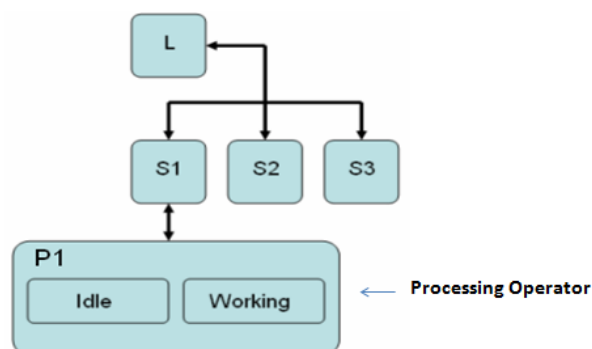


Figure 3.2: A functional example of an SRD

The Processing Operator shown in Figure 3.2 is a drill down into the service **S1**. The processing operator **P1** connects and relates elements to show the logical flow to construct a block



in order to illustrate the performance of a desired system. The performance of a system is illustrated in two foremost ways, expressed by the combination of communicational links and conditional operators. A processing operator can have several inner processing operators related to each other directly or indirectly; one or several of these processing operators can be used to serve some other block(s) of the system on a simultaneous basis. Let us assume the notations as shown below:

- L = List of Services
- S1 = Service 1
- P1 = Process Operator 1
- P1.1 = Idle Service
- P1.2 = Working

L is a **list of services** available in the Cloud containing a combination of **S1**, **S2** and **S3** (the **services** provided by a service provider) and **D1** (a service provided by a datacentre in the Cloud). There can be two major processes a service can provide, which are 0 and 1. The service is **Idle** or **Working**. The service might be idle due to no job being needed to be processed at this moment in time. The working service might also be idle, as there can be a delay in receiving some messages for other services or a datacentre. P1.1 (Idle) and P1.2 (Working) can further be drilled down. The Figure 2.19 shows a prototype of getting required information for the service user of **L** from the datacentre service **D1**.

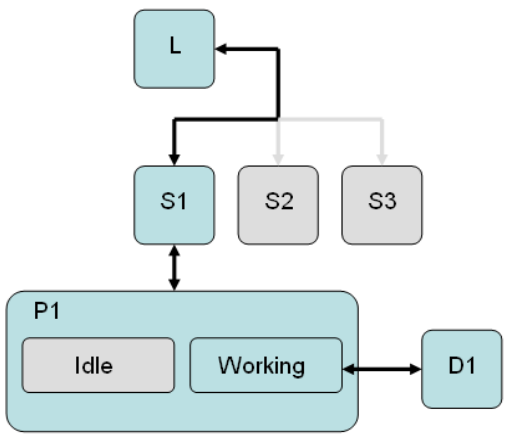


Figure 3.3: A working service prototype

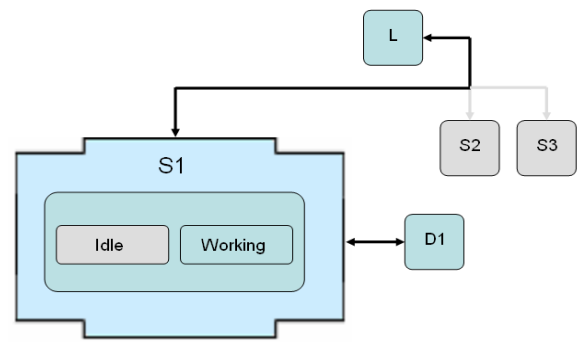


Figure 3.4: Another pictorial view of the SRD

The service **S1** as shown in Figure 3.4 is the actual service built as an application block. This service block **S1** connects with another service, **D1**, to get the required data for the user of service **L**. The processing block as shown earlier can be seen as an independent processing operator connecting two services by communicating the requestor's requirement, and the operator gets the resultant data set(s) and delivers it to the requestor.

Let us look into further details of these services; **L** is a listing service, we can consider it out of several services provided to stakeholders. For simplicity's sake, we will take the service **L** as given below:

- Detailed display of services available by a provider
- Where **L** facilitates users' request as input to get a resultant data set(s)

The service **S1** provides the following features:

- It generates a query on the request input
- By storing the data for future feedback for service designer
- It also manages bandwidth rates of data set(s) receiving from datacentre
- By delivering the data set(s) to service **L**

The service **D1** provides the following features:

- It receives query from **S1**
- Processes the resultant data set(s)
- Delivers to **S1**
- Stores the query for future use
- Stores query snapshot for feedback for service designer
- Self-manages the storage usage
- Manages bandwidth rates of every query

In SRD, a database is also considered a service. Here are a few real-life examples given below to understand the use of database services to the users on an everyday basis:

- An owner/operator of a transport company can use the service **S2** to enhance business by maintaining and adding the data associated with customer organizations. This database service will keep a record of load pick-up and delivery dates for all the customers (organizations) provided by the transport company. If this transport company is planning a special pre-summer promotion, the database service will be able to generate results of the customers to target for advertising to increase business.
- A chief organizer of ABC party campaigning in an election can take advantage of the same database service **S2** for a fundraising event. The mailing list generated by the database service is made up of a wide range of organizations with contact person's data of every organization arranged as per the area or city, town or state. These people can be invited with reference to the transport company's owner for a fundraising dinner for the ABC party for the election.

In the case of a merger between the manufacturer of a certain product, such as a printing company, and the transport company, the service S2 can be used to inform customer organizations of the newly available services of printing for clients to go with the transportation services as well.

Several business processes and service design methodologies have been developed and introduced in the last decade or so. Lin *et al.* [48] and Korherr *et al.* [49] provided comparisons of these methodologies in several surveys. Several designers at different organizations used UML notations and few used some [50, 51, 52, 53] special notation approaches. Event-driven Process Chains (EPCs) [50] is another prominent approach for requirements modeling for business process designing purposes. This type of modeling represents the dynamic performance of activities associated to specific business process requirements by focusing on depicting control flow

dependencies among several different process functionalities. There is a drawback using EPCs, and that is the extension of the user's provided requirement information creates havoc, where EPCs already have stored static information, as there is no such support provided to deal with this information addendums.

Kim *et al.* [54] worked on conceptual modeling and Kramler *et al.* [55] have produced work on web service collaboration protocols for inter-organizational business processes. Due to the use of SOA and related methodologies web services and XML have gained significance alongside of JSON. The use of XML has brought to the design world of software engineering XML-based notations. This new set of notations is being utilized to implement business processes modeling designs development. As per the work done in [56, 57] Web Services (BPEL4WS) and as described in [58] Business Process Specification Schema (BPSS) are the most trendy languages in the area of business processing.

Object Management Group (OMG) [59] introduced a new approach in 2006, and called it Business Process Modeling Notation (BPMN). This approach introduced a single graphical modeling notation for all stakeholders involved in a requirement design. The research work done by Karhunen *et al.* [47] provides us the study of a framework for SOA, which creates a well-defined business case using UML and BPMN diagrams containing business processes and the business requirements.

Knowledge management captures the individual knowledge, organization, and understanding of past processes and provides a model to logically justify the artifacts. The information exploration part plays a pivotal role in program understanding; it uses the information

structure generated by the knowledge management part and filters it according to specific required criteria. In this phase non-trivial program understanding of the system is acquired.

ASOA or Achievable SOA [17] is an approach that provides an efficient and cost effective methodology for organizations to achieve their required services fast and easy data gathering activities, and designers can design such applications using ASOA. This chapter puts light on the adaptable approaches, which can be adapted for software design of required services' front-end applications. It is very important to know the application design methodologies, such as abstraction, modular application design, and information hiding used to understand the available software structure. This understanding can be utilized as the basis to develop wrappers for legacy systems to convert data into XML/JSON for the use of services design.

The abstraction used in creating software designed with structured programming in mind did not focus its consideration on maintainability or application reuse as a service for a long term basis. The focus was instead placed on the application at hand to resolve an issue for a certain timeframe.

ASOA introduced a new paradigm in this chapter for the SOA solutions; SRD or System Requirements Design methodology. The human mind can do chunking within seven plus or minus two things at one time. ASOA-SRD contains eight diagrammatic elements to draw a design with. Every ASOA is to begin with an "Initiation Marker" and completes with a "Service (Block)". A service block is a simulation of a physical building block and can be reused for any other services' combination project. A stakeholder can be a consumer, user, or member of SOAT (Service Oriented Architecture Team).

A service block is created by using different service requirement elements. It is analogous to a physical building block, which is usually prepared with a cement and water combination,

where cement is made from lime, silica, alumina, iron oxide, and gypsum and it is a known fact that water is a combination of two elements Hydrogen and Oxygen. ASOA is a methodology that takes motivation from our physical environment.

After getting a snapshot of ASOA-Adaptable approaches with new and novel pictorial representation methodology, we move forward to get information of a novel ASOA-Approach, which is beneficial as being cost effective and time saving to design, develop, and deploy the service needed by any or all stakeholders of an organization.

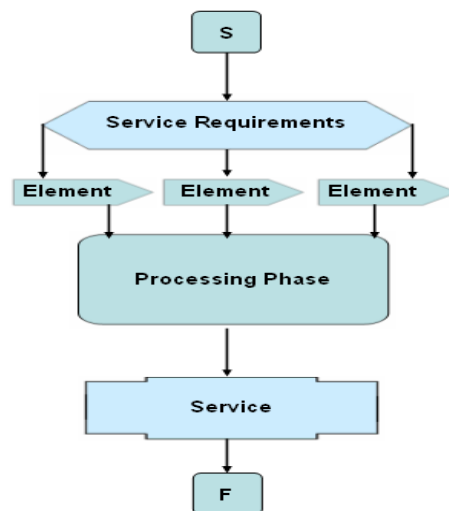


Figure 3.5: A general ASOA Solution Model

The start marker “S” is the initiation of the ASOA Service Solution designing. “Service Requirements” in the DFA (Demand for Alteration) are broken into further “Elements”, which are analogous to the earlier given example of cement. These elements are sent for processing in the “Processing Phase” to generate the required “Service”. The ASOA project is finished at marker “F”.

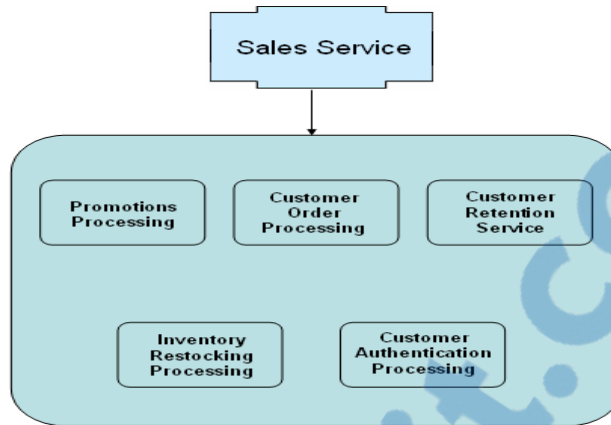


Figure 3.6: A drill down to processing phases of an ASOA solution

As shown in Figure 3.5 the service is designed using SRD pictorial representation. This SRD methodology gives us a zoom-in view of the processes, or processing phases, the consumer's request has to go through before an order is shipped to the requestor.

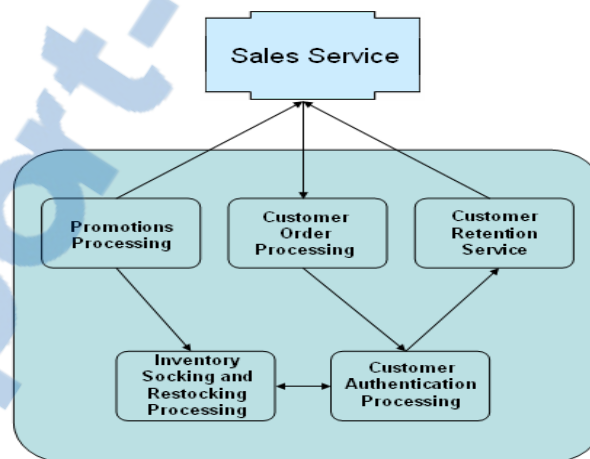


Figure 3.7: A detailed process view in one central process

A required services project can be considered as a combination of several web applications interconnecting several systems of one or many organizations. These designed services

individually receive input in the shape of some data or information and the service dispatches this request after a predefined conditional processing, if any, to get the requested results as an output for the consumer. An actual Service Block is shown in Figure 3.8. We can consider the service's design and development as a software system's development framework. Any of such design needs to use Requirements Engineering techniques [60, 61] to get specifications in the shape of stakeholders' provided requirements. These requirements of service design and service functionalities should be managed and precise. In the case of a SOA service design, the communication among the service designer team and stakeholders is vital to achieve the desired service quickly and efficiently.

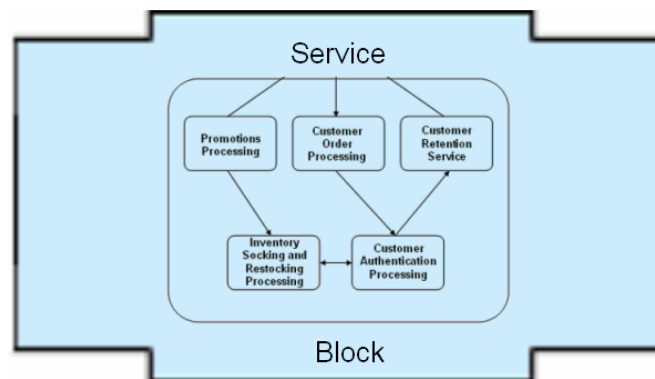


Figure 3.8: A Service Block

The goal of ASOA methodology is to develop and enhance the interactive relationship among SOAT and all non-SOAT stakeholders to achieve the desired service design, developed and deployed with accuracy and efficiency. In the world of SOA, it is a possibility that the end user is not a consumer or even an operational user; it can be another service requesting some data to deliver to another service at some other end used by a consumer connected via the web portal.

3.2 An ASOA Designed Service at Work

Let us see a functional service as an example designed using ASOA methodology with an SRD diagram in the given figures. The design of an application is of an ABC-CPP consumer products merchant offering general merchandize such as tissue papers, soaps, shampoos etc., to sales partners. As the main products provider ABC-CPP has to manage sufficient inventory available for fast delivery to its sales partners. As the stock level of a certain merchandize falls below a certain quantity, our central provider has to ask the manufacturing partner to ship the required product for restocking purposes. This transaction among all stakeholders can be understood as B2B2C.

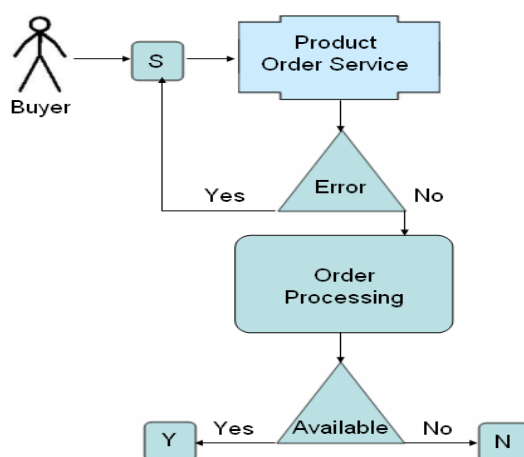


Figure 3.9: A consumer's order to buy a product.

Figure 3.9 shows a request to buy a product from a sales partner of ABC-CPP. In the case the ordered product is available at a retailer service, and another service provider handles the shipping, a B2B service will be invoked as shown in Figure 3.10.

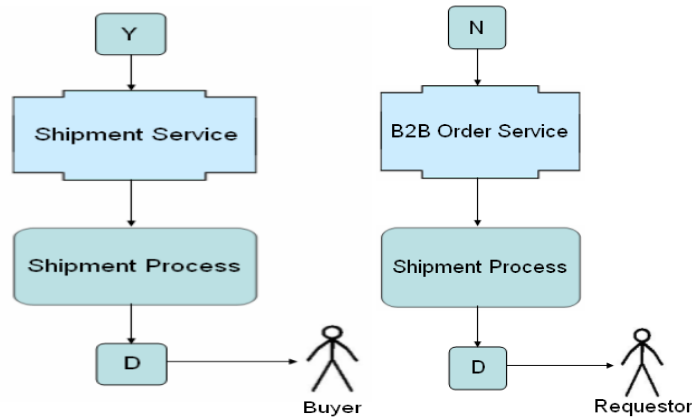


Figure 3.10: The completion of purchasing and shipment process

Figure 3.10 also shows the use of another service between the ABC-CPP and the manufacturer for restocking purposes or a B2B service. Let us drill down more into Figure 3.9, by looking at the Table 3.2 shows further processing happening in the given figure.

User	Request	Result	Progress
Buyer	Orders product	Correct request input	Ship the order
		Wrong Spellings	Error, help provided to correct the order
		Invalid product code selection	Error, help provided to user to correct the order

Table 3.2: Sequence of actions of a buyer

The solutions designed using SOA are disseminated and can be highly scalable. The ASOA approach is a novel methodology to design, develop, and deploy the service designed using available resources. These available resources can be legacy systems or distributed systems in use. ASOA has enormous potential to get pervasive recognition to achieve efficient and cost-effective

services. ASOA provides added methodology to all core components of SOA and a desired service can be designed, developed, and deployed for either one organization or by several businesses altogether to join forces to expand the services provided to the consumers.

Another core component of SOA is the ESB or enterprise service bus. ESB is a fundamental base communications infrastructure that serves the messaging among services; these services can be both front-end and back-end. ASOA tends not to use ESB. The messages are processed in the processing phase and are communicated among services using pervasive communication links. Figure 3.9 shows a complete model of all deployed services by proposed ASOA methodology. The security of services is discussed in the last section.

3.3 ASOA Solution's Progress Monitoring

There are several contributing factors attached to a successful ASOA [75] [76] service solution. The most crucial factor is to establish an efficient ASOA service solution's "Progress Monitoring System," or PMS, at the time of service design. The Progress Monitoring System needs to be designed to document the service development achieved progress milestones. This system will not only enhance the probability of achieving the desired service but will also advance a qualitative collaborative environment for all stakeholders involved in as SOAT (SOA Team) with an intense concentration to get the service deployed in the estimated time efficiently. This Progress Monitoring System is getting predominantly significant, due to the advancement of both hardware and software technologies.

The objectives of an effective "Progress Monitoring System" (PMS) are to get the service requirements' specifications prioritized (sequential outlining) with the scope of deliverables as desired service to enhance the benefits of users by being cost effective. PMS should have defined

milestones using an enforced design methodology. The specified elements of requirement specifications should be considered as identified jobs to be completed within some estimated time frames. PMS should also have an acceptance criterion agreed upon by all stakeholders as a milestone to achieve for testing the Beta version of the service.

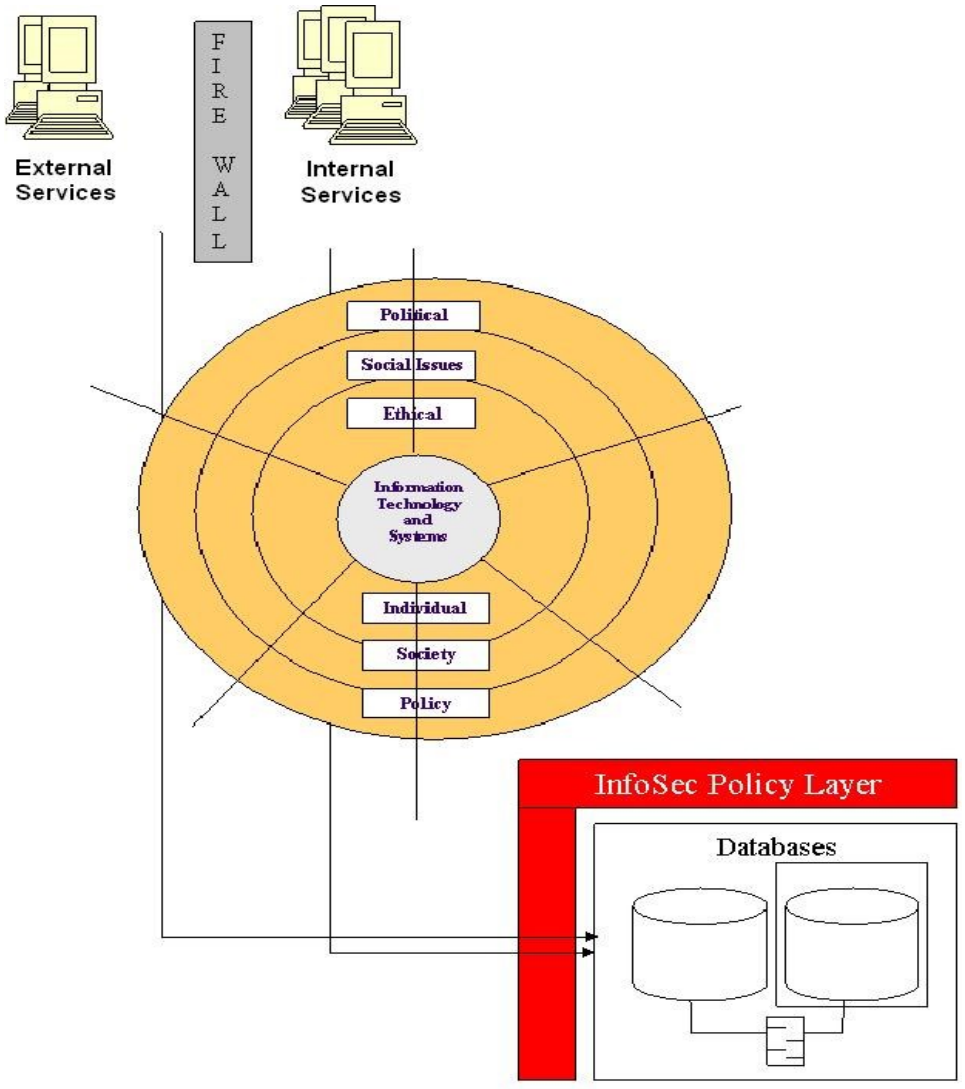


Figure 3.11: A complete ASOA design of secured services

SOAT has to establish as a part of PMS documentation, the effects of any assumed or non-assumed change in any of the organizational decisions and change of any requirement(s) during

the development time of the service to log and follow up on new, added, or altered modules of the desired service. PMS can make SOAT to deliver an efficient and cost effective service consistent with the requiring stakeholders' ideas and concepts. Figure 3.11 shows a complete ASOA solution, after tackling political, social, and ethical issues of all of the directly and indirectly involved stakeholders.

3.4 ASOA and Critical Factors

The rationale of ASOA and its associated classifications in the methodology discussed in this work is to define an efficient and cost effective service with an architecture designed and developed independent of available technological platforms. The choices of the available technology need to be determined wisely and should be based upon the business priorities without disrupting current serving status to the consumers and internal operational users.

The organizations must consider the given criterion in Table 3.3. These mentioned constraints with current issues must be carefully evaluated to achieve efficient and cost effective service for a fruitful business perspective. The skilled IT/IS managers and software engineers should find a complete consensus on legitimate basis before they initiate a service design. Cost effectiveness of an ASOA service solution should always be a top consideration to enhance the business profitability.

Constraint	Current Status	Predicted Status
ASOA Service Cost	Operational Cost	Reduced Operational Cost

Utilization	How transparent is the provided service(s)?	Ease of use
User Status	Users have to jump in different platforms to get data and do processing	Users will have one front-end to resolve their data processing issues.
User Trainings	Users need to learn several technology jargons	User will use a simple web portal in an available browser
Consumer's Patience	Consumers have to make phone calls or visit stores to buy the products need more patience	Consumers will get their desired products information on hand with available promotions and need not to wait and rethink of their buying decisions due to a store sales clerk's attitude, etc
Service Response Time	More time for operational users to serve their departmental requests or to provide several reports to get consumer's order delivered	The service will inform the consumer about product availability, shipping choices and delivery delays in advance. It will be a faster and prompt response to save consumer time
Secured Access	Users have to jump in between several portals to collect data for reporting purposes, any of the portal, if not correctly closed can be vulnerability	Service will hide all legacy systems under one umbrella and will auto disconnect, after getting user's required data

Table 3.3: Advantages of an ASOA Service Solution to an organization

3.5 ASOA for Cloud Computing

The challenges associated to SOA are presented in this thesis. We will drill down a few of these challenges, where ASOA can provide help to users of Cloud Computing. The challenges we will be looking at are:

- Flexibility is required
- Increasing demand of consumers to have standardized services with a seamless experience in their daily lives, such as users and their data communications
- It is the most important factor for any business to reduce the operational cost of these services by getting satisfactory service results by improving the efficiency, which means users can concentrate on controlling their business rather technology

ASOA provides model designing, which is flexible. Cloud computing provides scaling and transparent data communications to the users, as we have witnessed in the case of Amazon, Microsoft or Google Cloud. These providers maintain unique features for their consumers/customers and the provided features pose no issues in terms of technical problems for the users, rather they can spend more time conducting their business activities. The Cloud Computing is composed of many types of hardware, software and various operating systems.

ASOA provides (organizations) users pursuing their natural desire to work together to solve their widespread issues and their transactional needs. It is also a known fact that we live in the world of complex human relationships. The use of Cloud computing is based on mutual symmetric trust among service users and providers. Service requirements tend to change with the pace of business needs.

The service providers using Cloud can develop a large body of web services that can be distributed among service users with acceptable service level agreements (SLAs). The utilization

of Cloud computing is on the increase with the pace of time. The use of ASOA within Cloud computing is a new discipline, which introduces new fundamental concepts, where the need of service designers and providers is on the rise.

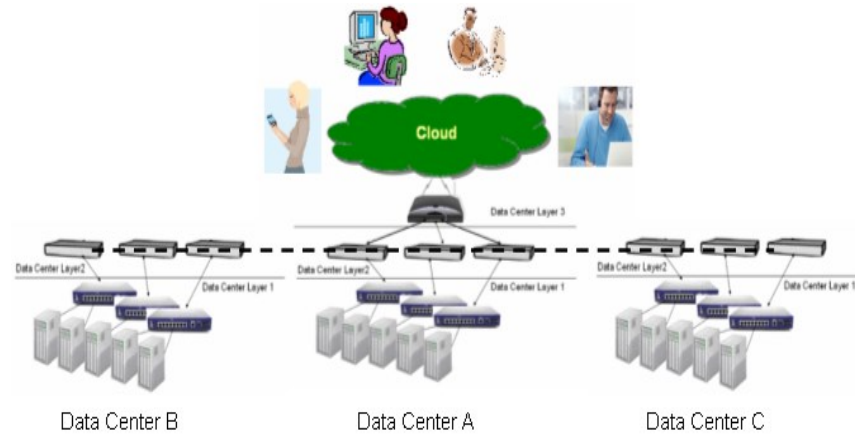


Figure 3.12: A combination of services for consumers

ASOA solution as mentioned earlier contains three vital stages.

- Data center layer 1 contains the data processing and storage servers, connected with each other for efficient and fast data transmission, eg. Hadoop/HDFS within datacenters.
- Layer 2 contains smart routers to balance the incoming request load [62] for layer 1 processing with firewalls to provide security to the communications.
- Layer 3 contains request receivers and response broadcasters.

3.6 Summary

This chapter has provided the details of pragmatic examples with the use of ASOA notations. The advantages of ASOA using the SRD as the base methodology to design systems for Cloud are also presented. Next chapter provides a secured gateway for Cloud in depth.

4 Cloud Computing Secured Gateway

The research conducted for this thesis also presents two novel algorithms. The first algorithm deals with Gateway Service, that is Cloud Server Algorithm (CSA), where as the second algorithm deals with the user or client side request processing for secured access. Both of the algorithms have been converted to mathematical form as well.

4.1 Cloud Server Side Gateway Algorithm

This Gateway Service is to receive requests to generate secured sessions for the user to work/process their data. A Block ID is used to prevent a threat from accessing the server. The database of blocked users utilizes a time sheet schema, which ranges from x min to the permanent blockades to potential threats. The database schema also has tuples that store such data like login and password. The login table schema individually stores the number of attempts the client tries to login and the use of the authorization key is also stored to generate sessions.

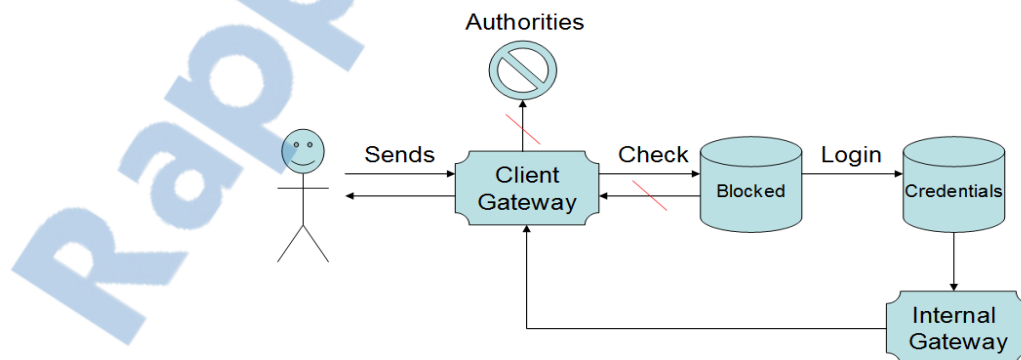


Figure 4.1 : ASOA Use in Secured Gateway

The Sessions table stores the log-in access, key, and ID. The Block table stores all ID's that have been blocked with locations with timer (it also contains areas that are blocked). Checking the ID at the gateway service means that the server will find the session ID and verify that it is the legitimate user, and also will verify that the ID is in a “safe network” (a trusted network where minimal risk of threats or security might be compromised). It also makes sure that another user isn't using that same ID.

4.1.1. Database Notation (SQL comparison): This research work uses notations like SQL Algebra, and it is given below, as the linear proof utilizes the given notations.

- **S** for Select or search query,
- **I** for insert new row,
- \emptyset for remove row and
- **M** for modify row.

4.1.2. Notations

- **MFSCChar - ::** indicates true path &
- **;;** indicates false path.
- Multiple of these gives a description of path needed to reach eg. **::;;::** = true false true,

An example is **Z::** should be read Z algorithm, First = true.

Pseudo Code	Cost(c)	Times
A		
Idle_State()		
Decrypt xc(Client_Request);	c0	dcx
If(IN_SESSION)	c1	1
Request_Client_ID();	c2	C
If (VALID)	c3	1
Generate_Page(DATA_REQUEST_1);	c4	D
ElseIf(DATA_REQUEST is Exit)	c5	1
Record_ID;	c6	1
Record_Session_Data;	c7	1
ElseIf(NEW_SESSION_REQUEST)	c8	1
Request_Client_ID();	c9	C
If (VALID)	c10	1
Create_Session	c11	1
In_Session_DB:		
ATTEMPT_NUMBER = new ATTEMPT_NUMBER,		
KEY = new KEY, INFO = CLIENT_INFO, ID = ID;		
Send(Encrypt_3s(LOG-IN_RESPONSE_SESSION_KEY);	c15 t	ec3 1
ElseIf(IN_LOGIN_SESSION)	c16	1
Process_Log-in_Request(REQUEST_1)	c17	B
Else//no validity to request	c18	1
Request_ID;	c19 t	1
Blacklist(ID);	c20	1

4.2 Mathematical Description: Cloud Server

The mathematical notation of Algorithm part A shows:

$$T = O(C) \text{ or } O(D) \text{ or } O(ec3) \text{ or } O(B) O(dcx)$$

Where **B**, **C** and **D** are also related parts to **A** and ec3, dcx are the time coding. The data access and requests require the Request_Client_ID() to prevent the compromise of any data and the log-in states it two times in the process to help prevent an agent from attacking with an interrupted request.

An idle state is the state of waiting for a request from a client in the network. It will first decrypt (part) of the request from the client. If the client is in an active session it will check the ID, then generate a page for the client (after security checks) (security check needs to be validated in order to generate a response). If the client requests to exit the session the server will record the

ID and halt the session (delete session from active sessions). If the client requests to login it will check the ID, if the ID is valid, in this case it will insert a new session to the login sessions database. Once this process is completed the login attempt(s) also send an encrypted log-in key to the user.

If the user is in a session the server will process the login attempt (Check_ID is in the login process). Finally, if the user is not of valid access, to the server, the server will record the ID of the Requested ID, and the request data. This will also notify the authorities about the threat attempt. The halt process will be executed with block ID storage.

$$\begin{aligned} & \sum_{\text{StartPartRequest}}^{\text{EndPartRequest}} dcx + \text{InSession} + \text{True}:: \{ \rightarrow \mathbf{C};; ; \rightarrow \mathbf{D}; \} \\ & \quad + \text{ExitSession} + \text{True}:: \{ \rightarrow \mathbf{IIDtoLog} + \mathbf{ISessionDatatoLog}; \} \\ & \quad + \text{NewLoginSession} + \text{True}:: \{ \rightarrow \mathbf{C};; ; \rightarrow \mathbf{InewSessiontoSessions} + \\ & \sum_{\text{StartSessionKey+response}}^{\text{EndSessionKey+Response}} es3 + tc; \} \\ & \quad + \text{InLoginSession} + \text{True}:: \{ \rightarrow \mathbf{B}; \} \\ & \quad + \text{NOTValidRequest} + \text{True}:: \{ \rightarrow \mathbf{C} + \mathbf{F}; \} \end{aligned}$$

B

Process_Log-in_Request(REQUEST_x)		
Decrypt_1c(REQUEST_x);	c0	dc1
If(SESSION_ATTEMPT < MAXIMUM_ATTEMPTS AND Parameters_Not_Null(c1	p 1
DECRYPTED_REQUEST_x))		
REQUEST_CLIENT_ID();	c3 t	C
Check_Log-in_DB(DECRYPTED_REQUEST_x);	c4	1
If (ACCOUNT_IS_VALID)	c5	1
If (ID_IS_VALID)	c6	1
Send(Encrypt_1s(ACCESS_APPROVED));	c7 t	es1
Remove_Log-in_Session;	c8	1
Create_Session;	c9	1
Else	c10	1
If (ID_IS_VALID)	c11	1
Send (Encrypt_1s (FAILED_ATTEMPT));	c12 t	es1
SESSION_ATTEMPT = SESSION_ATTEMPT + 1;	c13	1
Else	c14	1
Request_Client_ID());	c15	C
Blacklist(ID);	c16	F

The Analysis conducted in Algorithm part **B** shows:

$$T = O(dc1) \text{ or } O(es1) \text{ or } O(C) \text{ or } O(F)$$

$$\begin{aligned} & \sum_{StartRequest}^{EndRequest} dc1 + checkattempt \ 2SessionAttempts + \sum_{parameter1}^{parametern} ParameterCheck \\ & \text{True}:: \{ \rightarrow + C; ; ; \rightarrow + \sum_{i=1}^n \ 2parameter[i] Confirm + \\ & \quad \text{True}::: \{ \rightarrow ValidID + \text{True}::: \{ \rightarrow \sum_{STARTAccessApproved}^{ENDAccessApproved} es1 + LoginSessionfromLoginSessions \\ & \quad + \text{NewSessiontoSessions} + tc; \} \} \\ & \quad \text{False}:: \{ \rightarrow ValidID + \text{True}::: \{ \rightarrow \sum_{STARTFailedAttempt}^{ENDFailedAttempt} es1 + \mathbb{M}LogAttempts+1toLoginSession + \\ & \quad tc; \} \} \\ & \text{False}:: \{ \rightarrow + C + F; \} \end{aligned}$$

This part of the algorithm shows the decryption of the login request from the client. It checks if the number of attempts is less than the allowable number of attempts and if any of the fields are null. It requests for the client to provide their ID and checks to match a login from the database. If the account ID is valid (meaning it was used in a previous session or within the allowed ID areas) it will send the verification code exiting and deleting the login session (can also record the data accessed). This service simultaneously creates a session sending the client a key and other data encryption, while recording the ID and key. If the ID is not valid it will record the ID and block the ID halting the login session. This service will notify security of the organization. If the login attempt fails, it will simply send an error (encrypted) and add one more attempt to the session attempts. If the number of attempts exceeds the number of attempts allowed :

- The server will record the ID,
- Notify securities.
- Block ID
- Halt session.
- Basically blacklist the user.

C

Request_Client_ID()		
Encrypt_2s (Request_ID);	c0	es2
Decrypt_2c(ID);	c1	dc2
Check_ID;	c2	1
If(ID_HAS_CHANGED)	c3	1
Blacklist({ ID, ID_2 });	c4	E
return INVALID;	c5	1
Else If (ID_NOTVALID)	c6	1
Blacklist(ID);	c7	F
return INVALID;	c8	1
return VALID;	c9	1

The Analysis conducted in Algorithm part C shows:

$$T = O(ec2) \text{ or } O(dc2) \text{ or } O(E) \text{ or } O(F)$$

$$\sum_{STARTIDRequest}^{ENDIDRequest} es2 + tc + ts + \sum_{STARTIDResponse}^{ENDIDResponse} dc2 + 2SessionIDfromSessions + IDCompare$$

$$+ HasChanged \text{ True:: } \{ \rightarrow F + Return INVALID; \}$$

$$\text{False;; } \{ + NOTValid \text{ True::: } \{ \rightarrow F + Return INVALID; \}$$

$$\text{False;;; } \{ Return valid; \}$$

REQUEST_CLIENT_ID()

Encrypts a message to request the client ID

Decrypts the response from the Client then checks the ID

If the ID has changed within the session it will

 Blacklist the user and all IDs used.

 Returns invalid user

If the ID is not valid it will

 Blacklist the user

Returns Invalid user

It will produce the result of the ID check confirming or denying the ID is valid

D

Generate_Page (DATA_REQUEST_x)		
Decrypt_3c(DATA_REQUEST_x);	c0	dc3
Check_Account_Server_Restrictions;	c1	1
If(Approved)	c2	1
Generate_Data;	c3	1
Send (Encrypt_2s(CLIENT_REQUESTED_DATA_PAGE));	c4t	es2 1
Else	c5	1
Blacklist(ID);	c6	F
Send (Encrypt_2s(DENIED_ACCESS));	c7t	es2 1

The Analysis conducted in Algorithm part **D** shows:

$$T = O(dc3) \text{ or } O(es2) \text{ or } O(F) \text{ or } O(es2)$$

This will Decrypt the request then check the ID

It will then check the Account Restrictions

If approved to view the data from the request it will generate the data and send encrypted data to the client

If the client does not have authorization to view the data it will send an error message (encrypted)

$$\sum_{StartRequest}^{EndRequest} dc3 + \mathfrak{B}AccountrestrictionfromAccounts + CheckRestrictionApproved +$$

$$True:: \{ \rightarrow$$

$$\sum_{firstTransaction}^{LastTransaction} \mathfrak{L}something \text{ or } \emptyset something \text{ or } \mathfrak{I}something$$

$$+ \sum_{StartResponse}^{EndResponse} es2 + tc; \}$$

$$False;; \{ \rightarrow \mathbf{F} + \sum_{StartDeniedAccess}^{EndDeniedAccess} es2 + tc; \}$$

E

Blacklist(ID_x[])		
For(i = 1 to ID_x.Length(); i++)	c0	i
Blacklist(ID_x[i])	c1	i
End_Session;	c2	1

$$\mathbf{T}[] = (\mathbf{n} * \mathbf{F}) \sum_{i=1}^{\#ID} \mathbf{F}[i];$$

F

Blacklist(ID_x)		
Record_Prohibited(ID_x);	c0	1
Block(ID_x);	c1	1
Securities (ID_x);	c2	n
Record_Session_Activity();	c3	1
End_Session(ID_x);	c4	1

$$\mathbf{T}=\mathbf{O} (\mathbf{n})$$

$$\mathbf{T}IDtoProhibited + \mathbf{T}IDtoBlock + \sum_{n=1}^n \text{SecurityMeasure} + \mathbf{T}SessiontoLog + \emptyset IDfromSession$$

Blacklist(ID_x[]) and Blacklist(ID_x) will block, record, and notify security of all ID's listed from the array ID_x and remove the active session from the database. This records the activity of

session, this should already be done by the database, but the database should note the session details (ID, what was processed and when, login time, logout time, reason for ending session) halting the session, or going through the end session process (rather it sends log to security of the organization using this solution).

4.3 Client Service Linear Analysis

A Service user ID is the SPID, IP, and MAC, which are computer IDs used for networking. It is important to know that some of the variables, such as ACCEPTED in the following write-up, are derived from a previous function or process mentioned in the algorithm. All incoming and outgoing messages are encrypted and decrypted respectively. Notations are shown by a number followed by a character 's' meaning Cloud Server, and 'c' meaning Client Service. Encrypt_#char(DATA) & Decrypt_#char(DATA) represent different encryption and decryption methods used in this program. The number (#) mentioned here is a different encryption and decryption. The character is the source of the encryption.

The Time Analysis is done using variables Cost ($c\#$), Times ($t\#$) (number of times executed), with a mathematical notation proof. Data travel time from Client Service to Cloud Server is represented by 't'. Times are marked with letters (other than are ranged representations) depending on programmer's preference, and will be noted in the table given. Capital characters represent the algorithms stated below. Encryption and decryption algorithms run between $\Theta(n)$ and $O(n!)$ and are marked as $e(c \text{ or } s)\#$ and $d(c \text{ or } s)\#$ the 'c' and 's' represent client and server respectively.

4.4. Mathematical Description with Linear Analysis: Cloud Client

We have used discrete mathematics and a summation notation to describe the process steps in mathematical form. $\sum_{Start}^{End} Process$ describes a process where there is most likely a loop and that starts at some point in time increments, and ends at the end of the data feed. In short it is a sum of the steps used for each data size. Whereas $t_s + t_c$ describe the time it takes for a message to travel from Client Service to Cloud Server, where the server does some processes to serve the Client Service's request. If it only says t_s or t_c and does not have a returning t_x it means that the process does not require a return from the Cloud Server (this could mean that the return message from the Cloud Server is processed by another process).

Time O(J): Connect to Cloud server (Hand Shaking or some sort of security encryption process if needed). Connection will verify ID (ID is SPID, IP, and MAC). The ID will be encrypted; Cloud server will confirm the ID as the first step, if it is valid. In the case that the client ID is valid, it will notify the user by login display, accompanied by the encryption process (example: a key exchange and a confirmation code (Daemon or a process that is not visible to user)). The client service side's generation of login reduces the server process time, it also aids in prevention of undesired access.

Unless the client will generate its own error page, **Halt** is meant to end the session and close the program.

grs + **E**

True:: { -> + **2**loginpage + printLoginPage;

False;; { -> + **2**Error page + printErrorPage + Halt;

T = O(ec) or O(ds): **p** depends on how many fields (or data inputs the user is needed to fill) are in the login.

B

Go_Log-in_Request()		
If(Log-in_Parameter_Check)	c0	p
Send(Encrypt_1c(LOGIN_NAME,PASSWORD));	c1t	ec1
Server_Request_ID(Decrypt_2s(Server_Request_1));	c2	E
Log-in_Response(SERVER_RESPONSE_1);	c3	C

Time = O(p). When the Client service user initiates the login request (by a button or some other method) the login name and password must not be null. (This will reduce failed attempts and also the cloud server will check for null fields and mark the user as compromised if it finds one. It will send the encrypted login name and password along with a session key, and then will wait for the cloud server response, and decrypts the received data for the next process.

Log-in_Response (SERVER_RESPONSE_x).

$$\sum_{p=0}^n p + \sum_{p=0}^n ec1 + ts + tc \rightarrow + E:: \rightarrow + tc + C::;$$

C

Log-in_Response(SERVER_RESPONSE_x)		
Decrypt_1s(SERVER_RESPONSE_x)	c0	ds1
If(ACCEPTED_LOGIN)// if the ID is valid to the server	c1	1
On_Log-in_Response(DECRYPTED_SERVER_RESPONSE_x);	c2	D
Else	c3	1
Print(LOGIN_PAGE);	c4	1
Go_Log-in_Request();	c5	1

T = O(ds): Decrypts the Cloud server's response, from the login message, that was sent. If the login is accepted it will process **On_Login_Response(RESPONSE)**. If not accepted it will go back to the login page.

$$\sum_{StartsResponse}^{EndsResponse} ds1 + Accepted$$

True:: { -> **D**;

False:: { -> **2**Error page + printErrorPage + **B**;

D

On_Log-in_Response(SERVER_RESPONSE_w)		
Server_Request_ID(Decrypt_2s(SERVER_RESPONSE_w));	c0	E, ds2
If (ACCEPTED) // Log-in Parameters are accepted	c1	1
Display_Session_Page(DECRYPTED_SERVER_RESPONSE);	c2	1
Else	c3	1
Go_Log-in_Request();	c4	B

T = O(ds) or O(E) or O(B): Will first decrypt the cloud server's response. From the server's response the client service will determine if the user is logged in or not, and if the ID is valid. If accepted it will generate a session page. If not accepted it will go back to the login page.

$$\sum_{StartsResponse}^{EndsResponse} ds2 + E:: + tc + Accepted$$

True:: { -> DisplayPage;

False;; { -> **B**;

T = O(ec): s are the other security measures.

Estimate T = O(1) to O(n): Will check to make sure the server is correct. If the key is correct it will send the computer's SPID, IP, and, MAC. If the key is not correct it will go through a series of actions, locking and halting the session, and notifying the user that the session is compromised.

E

Server_Request_ID(SERVER_RESPONSE_z)		
Check_Server_Signature_Key (SERVER_RESPONSE_z);	c0	1
If (CORRECT_SERVER_KEY)	c1	1
Send(Encrypt_2c (Fetch(SPID, IP, MAC)));	c2+t	ec2
Else	c3	1
Session_Locked;	c4	1
Request_Server_ID;	c5+t	ec3
Message_User(SERVER_COMPROMISED);	c6	1
Record_Server_ID;	c7	1
Send_to_Securities(Encrypt_4c(Server_ID, Client_ID, SERV- ER_RESPONSE_DECRYPTED));	c8+t	ec4
Extra_Security;	c9	5
Halt	c10	1

Requesting and recording the Server ID, while making sure to send a message to security (is the job of another server that will take action to curb threats), and it halts until the issue is taken care of. **Extra_Security** is something to the point that the program will not start again unless explicitly authorized.

CheckSignature

True:: { + -> 3ID + $\sum_{StartID}^{EndID} ec2$ + ts;

OR

False:: { -> + c4 + ts + t6 + tc + 1ServerID +
 $\sum_{StartSID}^{EndSID} ec4$ + $\sum_{i=1}^n SecurityOperation [i]$
+ Halt;

F

Request_Data(COMMAND_x)		
Send(Encrypt_3c(Generate_Data_Request(COMMAND_x)));	c0+t	ec3
Server_Request_ID((Decrypt_2s(SERVER_RESPONSE_3)))	c1	ds2, E
Generate_Session_Page(DECRYPTED_SERVER_RESPONSE_4);	c2	1

T = O(ec): Sends an encrypted message to the server, and the message is generated from the user's command. It will then process the server's request for the client user's ID. From the response the session page will be displayed. The server reserves the right to deny access to clients (Note: denial of access results in blacklisting of the client).

$$\sum_{StartDataReq}^{EndDataReq} ec3 + ts + tc \rightarrow \sum_{StartResponse}^{EndResponse} ds2 + PrintServerResponse;$$

G

On_Session_Exit()		
Send(END_SESSION_CODE);	co	1
Halt;	ci	1

T= O(ec): It will send an end session code to the server to terminate the session; then halts the program.

$$\sum_{StartEndSession}^{EndEndSession} e + halt;$$

H

On_Command()		
Request_Data(COMMAND_1);	co	F

T = O(F): It will send an encrypted request to the server then Request_Data(COMMAND_MADE). A simple example for an on-click or other GUI command.

See Request_Data(Command)

$T = O(ds)$: It sends a request for a login session to the server. The server will request an ID and the client will decode the response and determine if the server is valid. The client will then record the login key, this will prevent Man in The Middle (MITM) threat.

J

Connect_To_Server()		
Send(Request_Log-in_Session);	c0+t	1
Server_Request_ID((Decrypt_2s(SERVER_RESPONSE_1));	c1	E
Record(Decrypt_3s(Server_Response_Log-in_Key));	c2	ds3

$$\begin{aligned}
 & \text{GenerateRequestLogin} + \\
 & ts + tc + \sum_{\text{STARTServerResponse}}^{\text{ENDServerResponse}} ds2 + E:: -> + ts -> \\
 & \sum_{\text{StartSResponse}}^{\text{EndSResponse}} ds3 + \text{IDecryptedResponse};
 \end{aligned}$$

Hence the conclusion for time analysis is that operation time is dependent on Encryption and Decryption algorithms used ($T = O(n)$ best case and $T = O(n!)$ worst case), if we ignore Encryption and Decryption the runtime can be described as $T = O(1)$ in most cases for the algorithms used, and $T = O(n)$ in some processes such as **On_Login_Response()** which has a parameter check.

These algorithms are tested using a “Linear Analysis” mathematical proof and in the following chapter, concludes this research work and provides a novel Cloud Secured Gateway. It provides a working example that deals with a real world example of data security, governance, and challenges faced to secure data.

4.5 Cloud Security Gateway

The utilization of a Cloud Security Gateway can be seen in the given figure 4.2, to understand the Opensource customizing of the provided algorithms according to organizational internal requirements within policies to make sure to avoid any compliance issues within a State or Governmental jurisdiction. The embodiment of the processes shown in Figure 3.7 within Figure 4.2 shows the relationship, and how an organization can customise using Opensource technology within internal and/or external Cloud utilization.

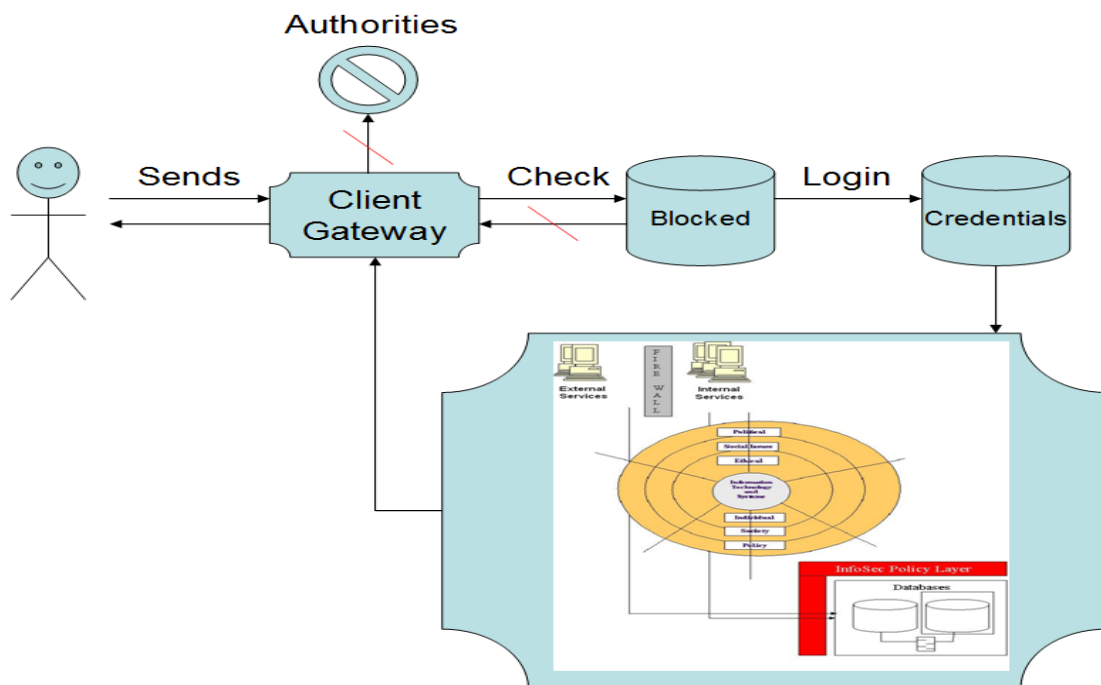


Figure 4.2: Cloud Security Gateway

4.6 Summary

This chapter has presented a secured gateway solution for the Opensource adaptation by any organization. Any industrial entity, which would like to customize their internal services Cloud security development to safeguard their electronic assets, can write their own Cloud Security Services using the algorithms provided in this chapter. Chapter 5 discusses the use-cases designed using the algorithms provided in this chapter.

5 Cloud Security Gateway – Use of ASOA

The advent of the computer age brought significant change into our lives. We use computers every day, almost everywhere. The business environment around us is continuously changing at a very fast pace as Cloud computing is taking over as a new business environment. To cope with the issues of process changes, such as adaptation of new or different rules, and regulations at the time of an expansion of a business, extends a need of its business processes to be reengineered. This thesis discusses three fundamental concepts of Business Process Reengineering (BPR), Systems Reengineering, and the extended use of Service-Oriented Architecture (SOA) in terms of Achievable SOA (ASOA).

5.1 ASOA Proposed Services Framework

This research also proposes a services framework, which can be adapted by the industry to get cost effective and efficient services for their internal users as well as for their customers. The example given in this thesis is based on actual business practices of a real estate business. The combination of several services provided by a consortium of real estate companies are shown in the Figure 5.1. These services coordinate with each other's within a Cloud and only one service has the user front-end. It does not matter, whether ABC Real Estate is using Windows as an operating system or XYZ Real Estate is using a LINUX.

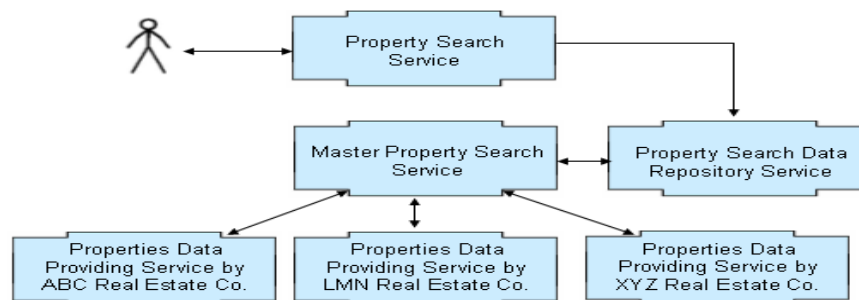


Figure 5.1: Combination of several services in a Cloud

5.1.1 Use of Map-Reduce in ASOA: Map-Reduce [63] is a combination of Map and Reduce functions. This combination was first successfully used for the Google search facility at Google Labs. There are giant datasets, which are also known as Big Data, sometimes larger than a Terabyte of data as an input request by hundreds or thousands of users. It is obvious that this processing clearly was not going to happen on one server or even on one datacenter as shown in the given Figure 5.2. It is going to take several hundred or maybe a thousand CPUs to crunch a Terabyte or Petabyte of data in a reasonable amount of time.

The research teams at Google wanted to make this search process involving several datacenters, easy and faster for users to find their needed information. This was the basic reason that MapReduce was introduced as a framework which provides this automatic parallelization and distribution to find document(s) requested by a user. The use of MapReduce is given below in the ASOA Service proposal for Cloud computing.

MapReduce provides fault-tolerance as one of the important features, as this combination of functions tries to access thousands of computers to deal with Big Data. It is a possibility that one of these computers might crash or have a faulty network card at any given time.

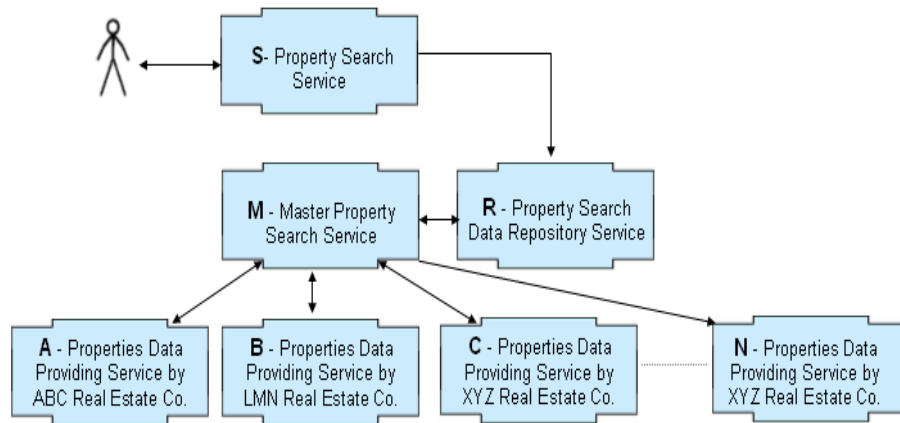


Figure 5.2: MapReduce in ASOA proposal for Cloud

MapReduce also provides status and monitoring tools in the shape of Big Data hosted address, where the correction is found in the data repository service “R” as shown in the Figure 5.2, so that data center administrators can be sent the information of found fault to get their server(s) fixed.

Figure 5.2 can also be depicted as:

$$S = R$$

$$S = R + M \quad \text{Where } M = \text{Sum } (A, B, C, \dots, N)$$

The Map-Reduce [62 64] model is given below:

Map (String key, String value):

//key: document name

//value: document contents

For each word w in value;


```
EmitIntermediate (w, "1");
```

Reduce (String key, Iterator values):

```
// key: a word
```

```
// values: a list of counts
```

```
int result = 0;
```

```
for each v in values;
```

```
    result +=ParseInt(v);
```

```
    Emit (AsString (result));
```

- It is obvious that MapReduce will work on a list of Key-Value pairs, so each map function is going to take an input key and input value, and is going to return one or more output values with their own (intermediate) values, with their own output keys.
- The reduce function takes all the intermediate values for a given output key and creates a list of final values that have been created by the aggregation through them.

The values that are fed into the mapper function are some kind of records from some data source(s), which may be some lines out of a file or some database rows etc., and each of these input values comes with some sort of an input key. The map is going to produce intermediate values along with an output key from the input. After the map phase is over all the intermediate values from all of the different mappers are combined together into a list for a given output key, which has a final value or a list of values.

5.1.2 Use of Parallelism: The parallelism that gets created by MapReduce comes from the fact that all of these different map() functions run in parallel and all of them are using totally different input datasets, since each mapper can't see the other input dataset there is no synchronization between them. These mappers just start writing output values and can run in different address spaces in the Cloud for all users at various locations.

The reduce functions can also run in parallel because the output keys are constrained to be working on separate datasets, the only bottleneck in this process is that the reduce function can't start until the map function is completely finished. In case one mapper is not able to produce a resultant dataset, the reducer will remain idle. However, this issue has been taken care by the Opensource Hadoop community by introducing YARN (Yet Another Resource Negotiator) in every slave node in the cluster, where we use MapReduce to distill datasets.

In ASOA proposed framework, MapReduce is used, if each shape [64] in the given Figure 5.3 is considered as a service provided by a service provider (a business) and a requester service or a user request to use a few or one of these services, the Figure 5.3 given will be the exact simulated replica of such a processing:

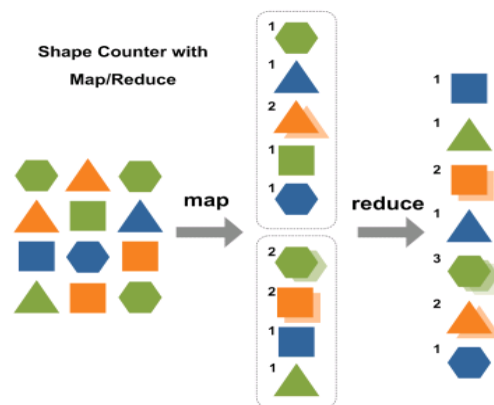


Figure 5.3: The use of services using Map-Reduce [64]

5.2 Use of Services in a Cloud

The use of Cloud computing provides a user (an organization) computing resources, which can be utilized by the user on-demand with the increase or decrease of computing servers, network bandwidth and storage, and these utilizations are subject to an service level agreement (SLA) among the user and the Cloud computing services provider. The cost can vary with the utilization increase and decrease. Figure 5.4 shows an internal service communication among several services within a Cloud.

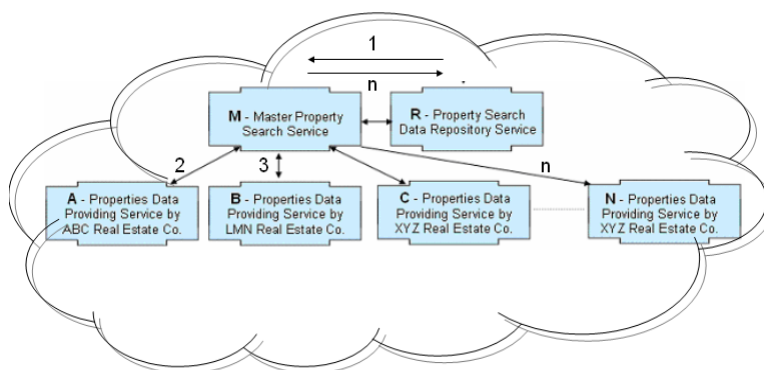


Figure 5.4: Service discovery paths in a Cloud

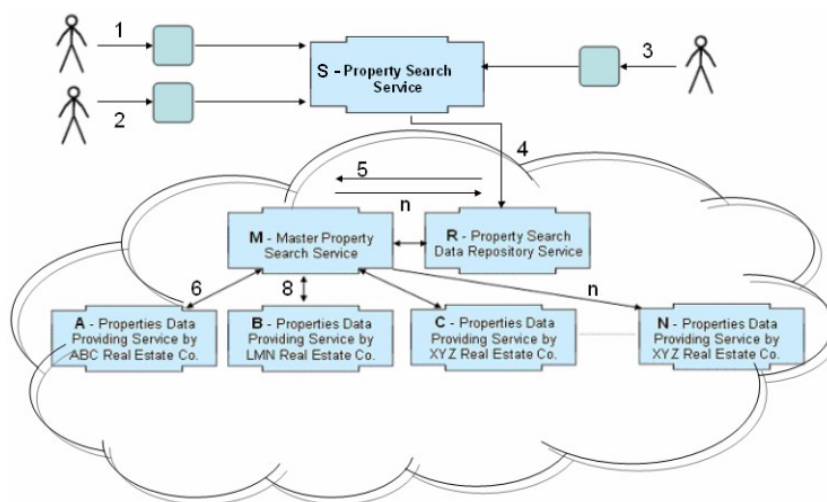


Figure 5.5: Requestors Servicing Input



The service **S** executes a search request for a property for sale on a repository **R**. If the requested data is not available service **R** generates a request as shown as path 5 to the Master search service **M**, which generates a query from the registered service with it as data providing services.

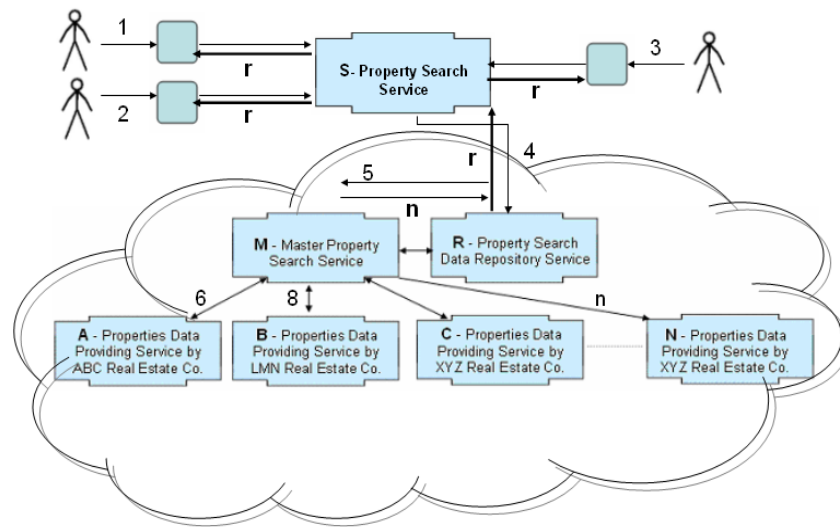


Figure 5.6: Requestors servicing output

The data repository service **R** delivers the requested data using return path “**r**” to the property search service **S**, if it is available from previous searched data. In another case, the master service **M** will find data and send it to data repository service **R**. Let us denote the diagrammatic notations as given in Figure 5.6, on the basis of the given figure, we can see the transformed Figure 5.4 as shown in Figure 5.6.

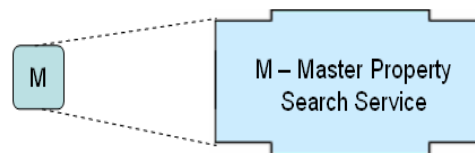


Figure 5.7: Notating Service M as a start/end service Marker.

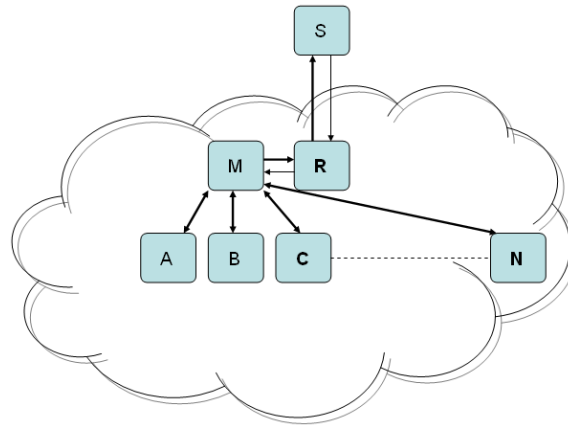


Figure 5.8: Services available in the Cloud

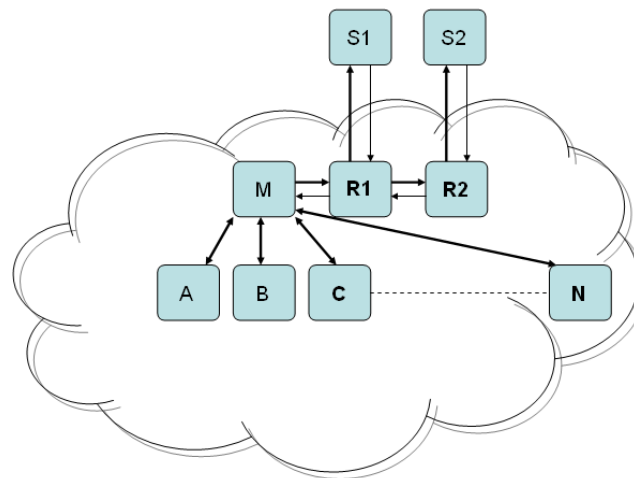


Figure 5.9: Utilization of two different services S1 and S2

In one case, if there is a second user willing to get information about some properties available for Short Sale, who uses another Service **S2**, which will find the data available from the Short Sale Data Repository of R2, the search transaction will be able to query repository R1. This might contain some properties, which might be on short sale and were searched by user 1 in a complete combination of all available properties on sale by several property selling companies. In case **R1** does not have any data the request will move to service **M** for further research. This example illustrates that a Service can be understood as a self-describing black-box like function,

which can be called by another service or an application via its URL address that will provide a resultant dataset in response to a request by a service user or by another service.

An interesting question can arise here, about scalability, as per service level agreements (SLA) between the providers of service **M** and **R1**, if the data should not be increased from a certain storage, let us assume 30 GB is the agreed limit, otherwise the charge of the service will rise as well or some older data should be removed to get new data stored for future availability.

5.3 ASOA Use Cases

Cloud computing and Big Data is the reality of our time and as we discuss BPR, we have also provided research on SOA, Cloud Services, and MapReduce. The velocity with which this Big Data is being accumulated in our contemporary business environments can be optimized by using Cloud Computing using BPR on a better note for an organization by utilizing Opensource solutions. There are no standard security protocols as of yet available in the world of Big Data within Cloud in the Opensource technology adaptations by the organizations, who would like to avoid utilizing Closed Source solutions, such as Microsoft, Oracle or IBM, etc. This thesis presents a Cloud Monitoring Gateway using ASOA, for secured transactions on Big Data for any analytical needs of the organization using Opensource technology.

Login - Use Case A: User will attempt to log in by sending valid credentials and server will respond with a successful login and rendering the requested page.

The following diagram is in SOA 3.0 methodology

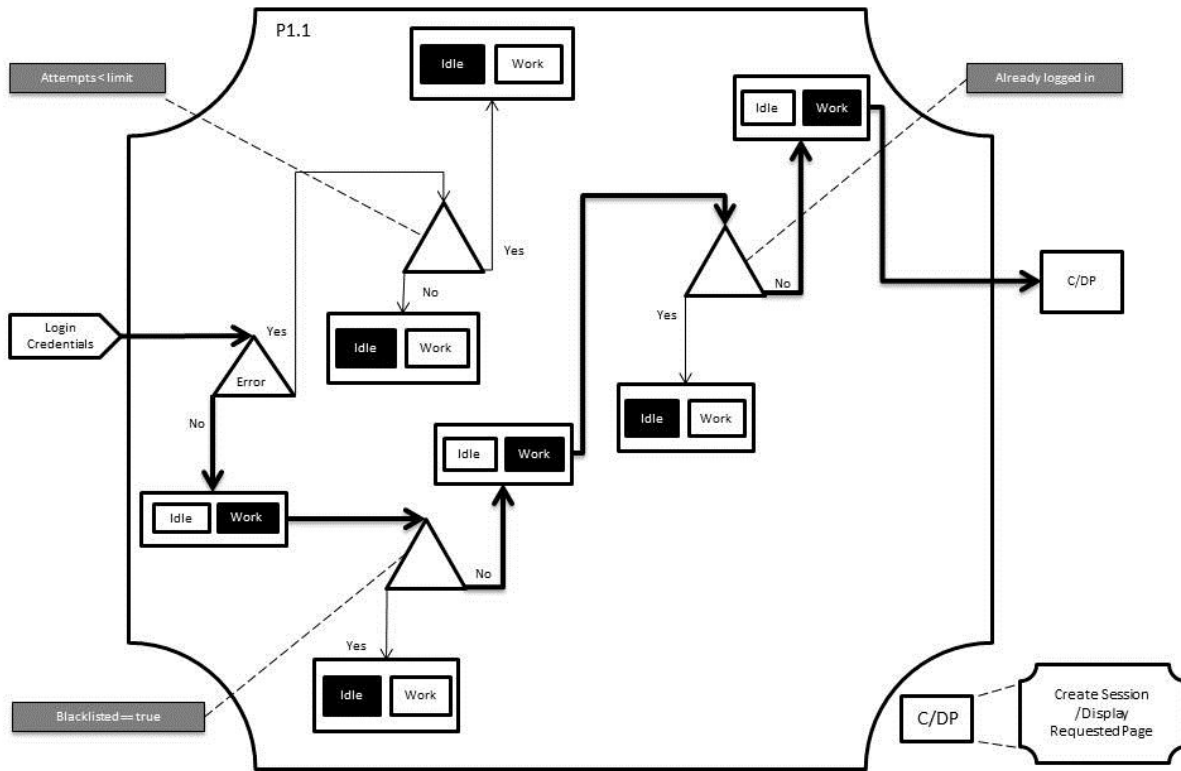


Figure 5.10: Login – Use Case A.

Login - Use Case B: User will attempt to log in by sending invalid credentials for first time and server will respond with a failed message and increase invalid attempt by 1.

The following diagram is in SOA 3.0 methodology

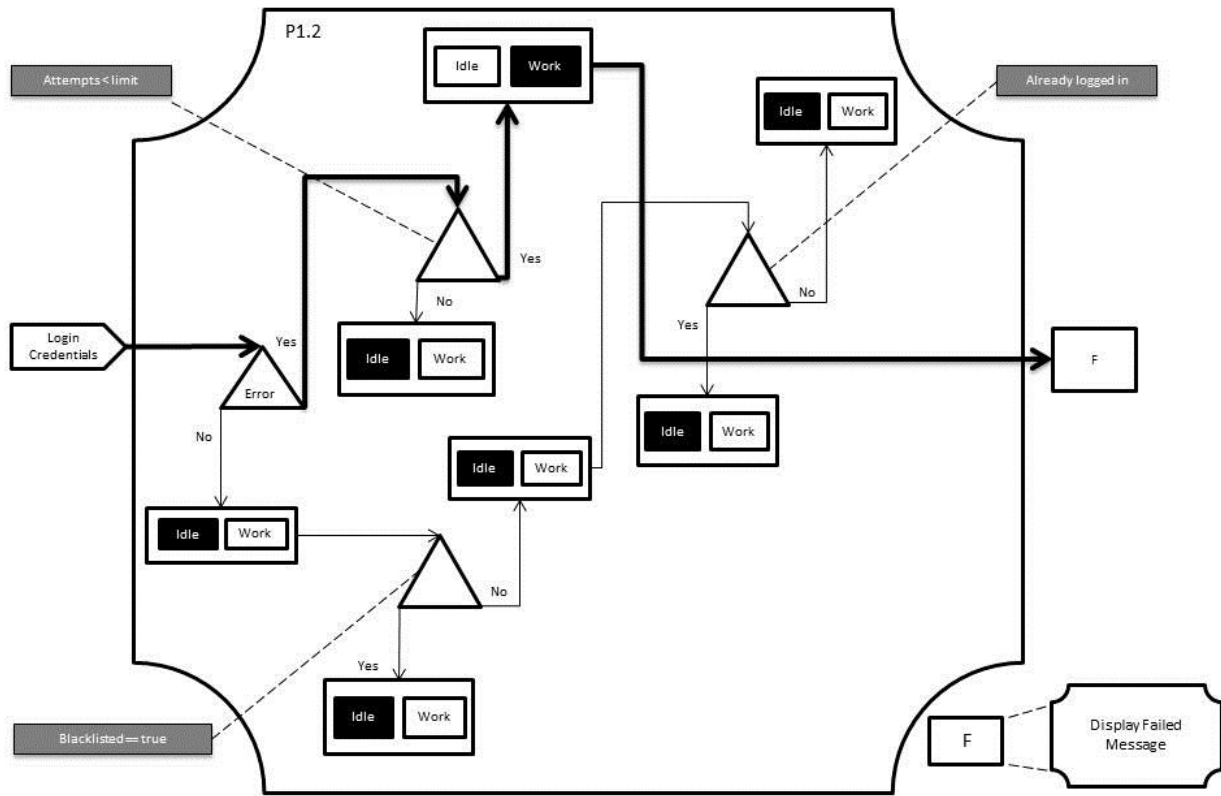


Figure 5.11: Login - Use Case B.

Login - Use Case C: User will attempt to log in by sending invalid credentials and invalid attempts over threshold limit and server will blacklist the id and end the session.

The following diagram is in SOA 3.0 methodology

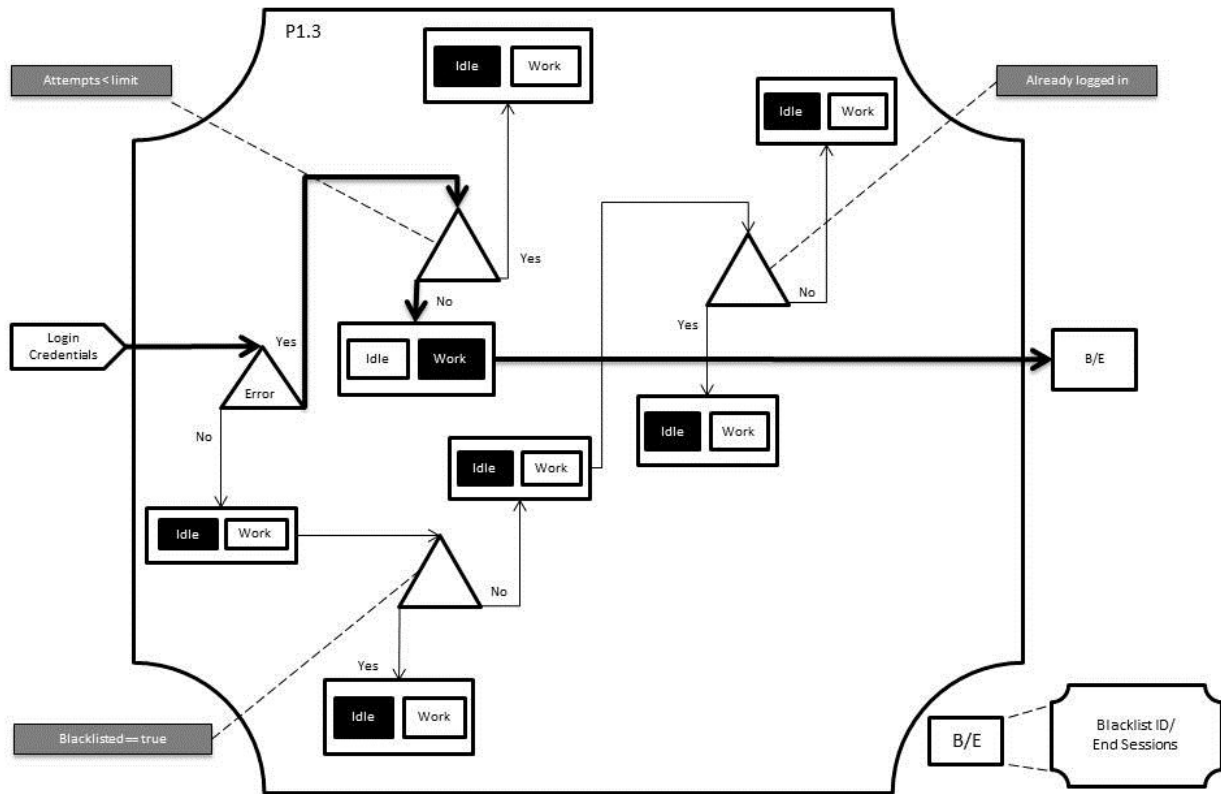


Figure 5.12: Login – Use Case C.

Login - Use Case D: User that is blacklisted attempts to login in. The server denies access

The following diagram is in SOA 3.0 methodology

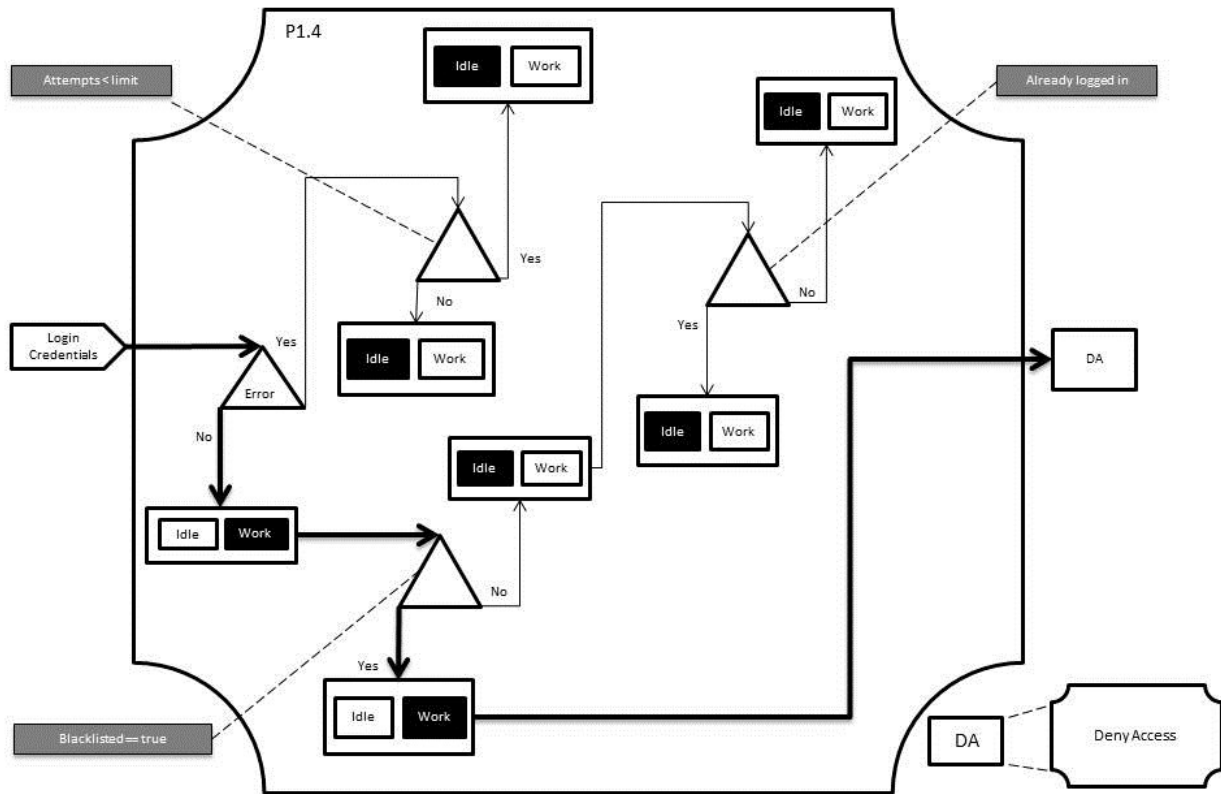


Figure 5.13: Login – Use Case D.

Login - Use Case E: User attempts to log in while logged in at another location. The server will invalidate the old session and create a new one.

The following diagram is in SOA 3.0 methodology

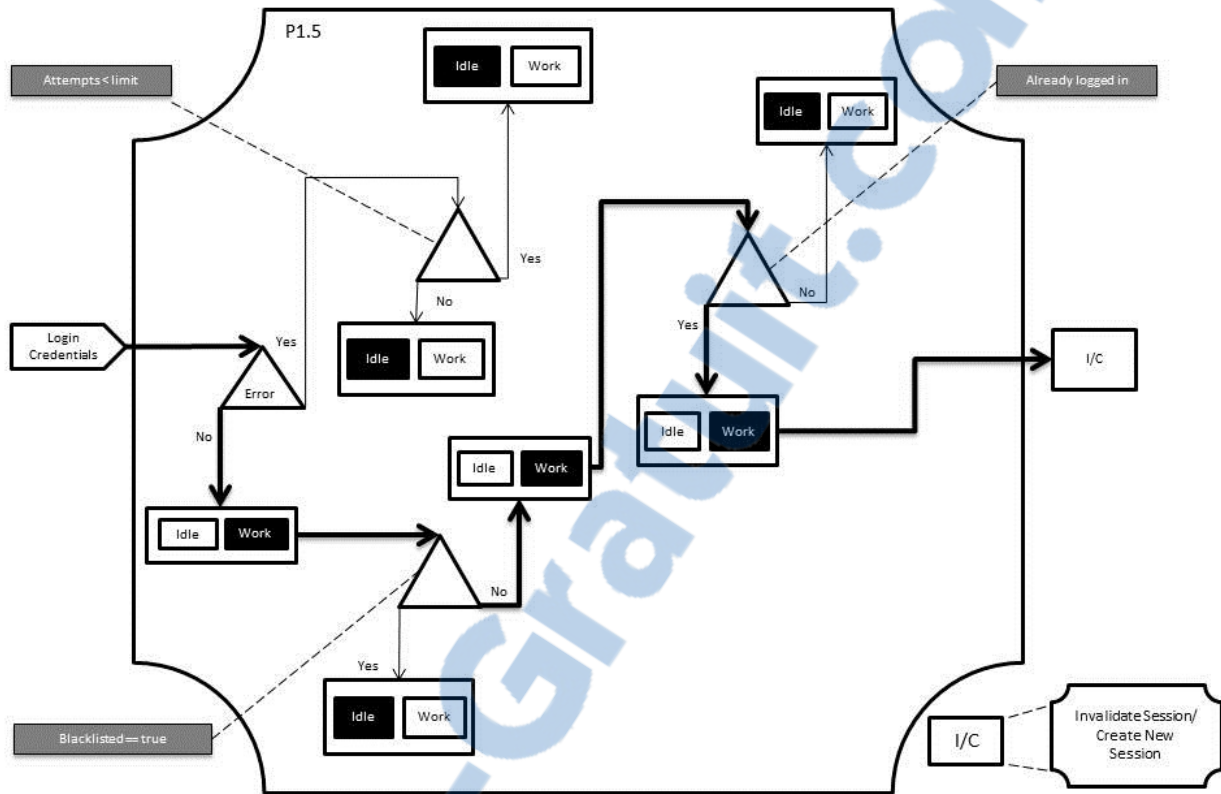


Figure 5.14: Login – Use Case E.

Session Change - Use Case A: Users IP changes mid session and a request is sent to the server. The server responds by blacklisting all id's used in the session and ending the session.

The following diagram is in SOA 3.0 methodology

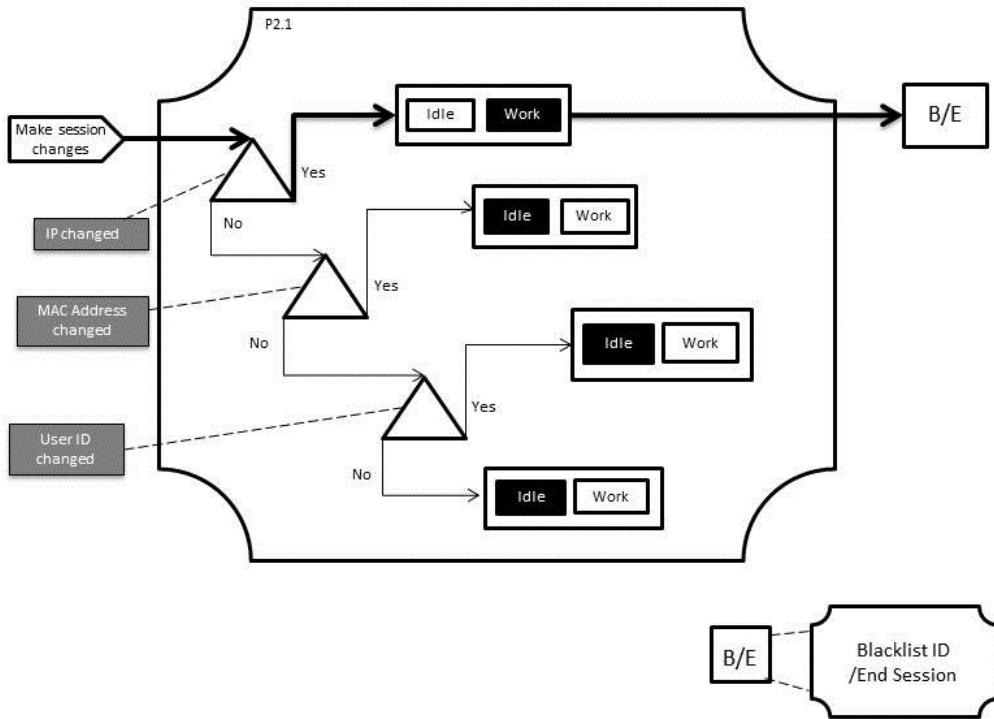


Figure 5.15: Session Change - Use Case A.

Session Change - Use Case B: Users mac address changes mid session and a request is sent to the server. The server responds by blacklisting all id's used in the session and ending the session.

The following diagram is in SOA 3.0 methodology

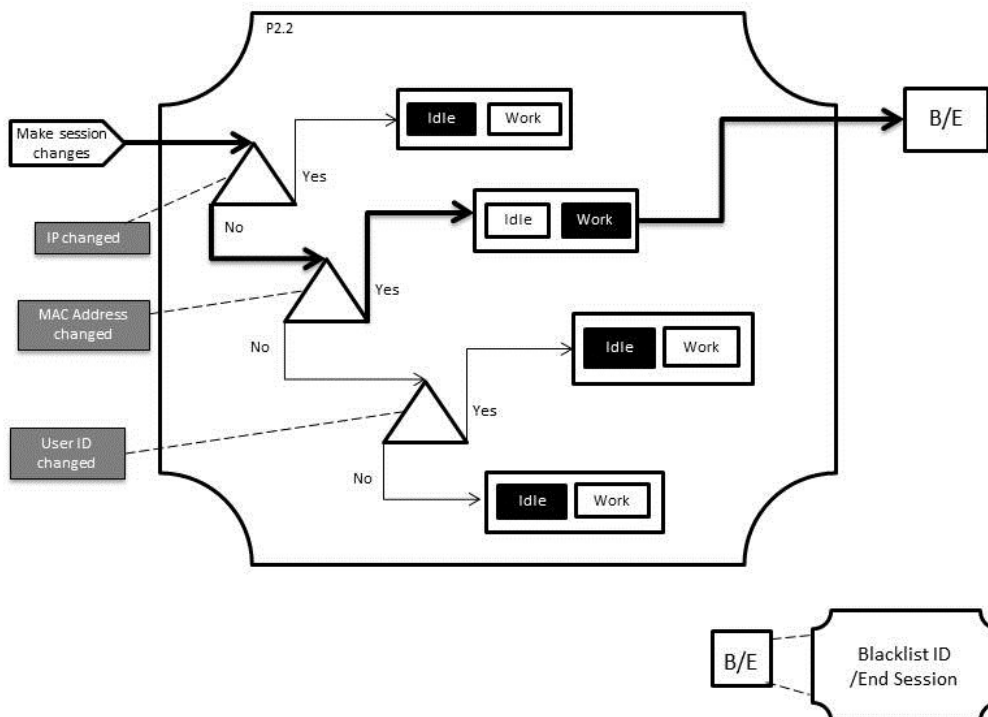


Figure 5.16: Session Change - Use Case B.



Session Change - Use Case C: Users user id changes mid session and a request is sent to the server. The server responds by blacklisting all id's used in the session and ending the session.

The following diagram is in SOA 3.0 methodology

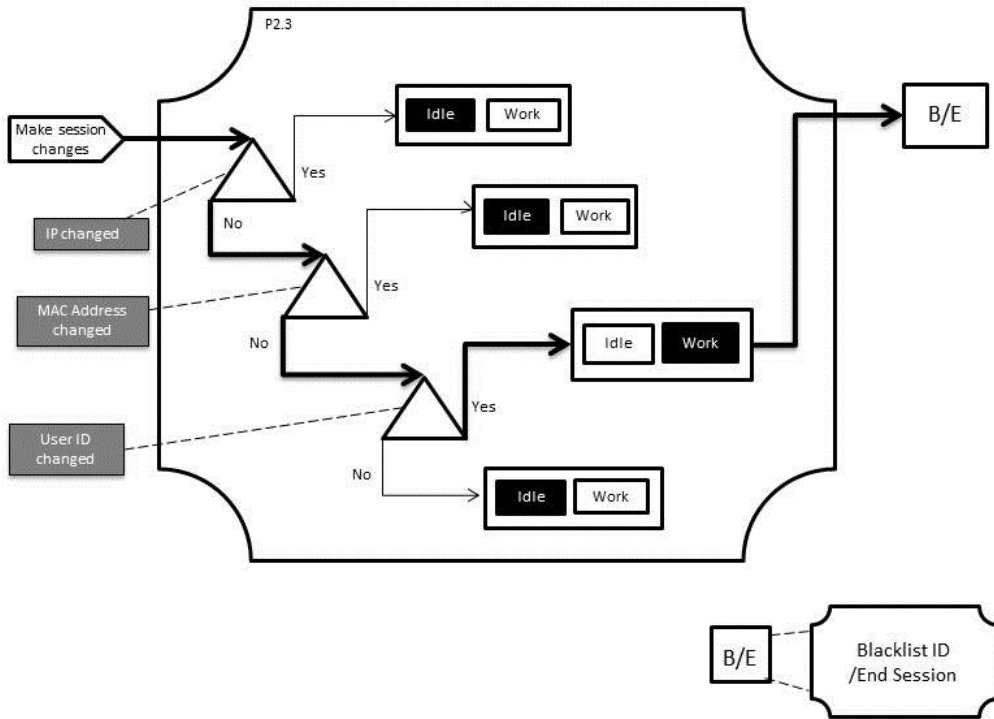


Figure 5.17: Session Change - Use Case C.

Page Request - Use Case A: User is already logged in and requests access to a page he has access to. The server responds by sending the requested data.

The following diagram is in SOA 3.0 methodology

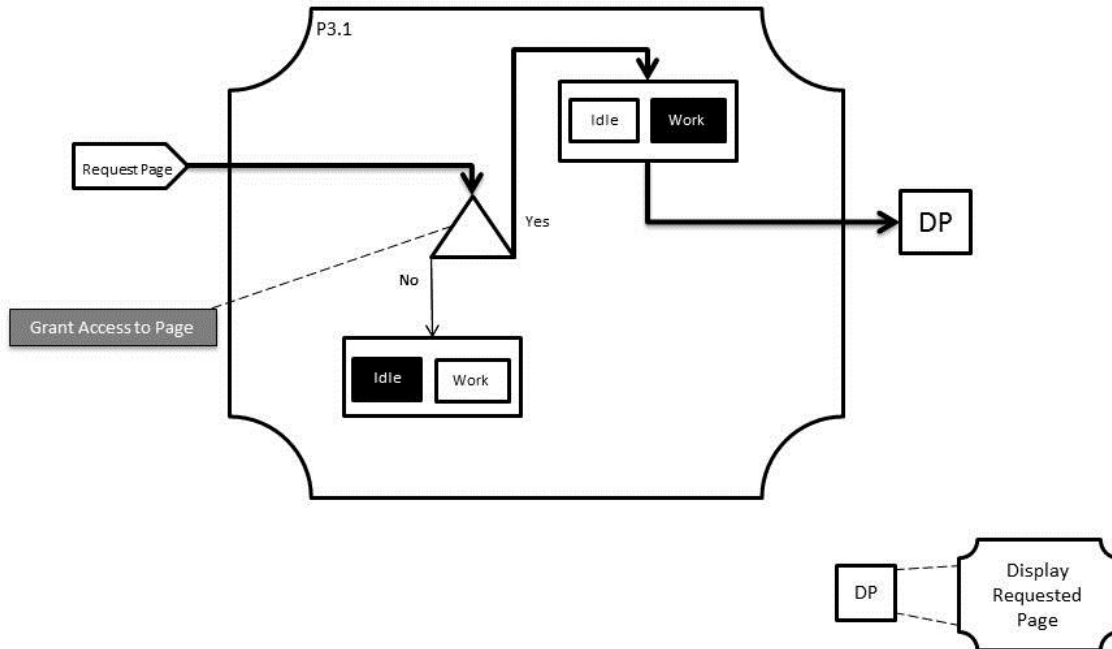


Figure 5.18: Page Request - Use Case A.

Page Request - Use Case B: User is already logged in and requests access to a page he does not has access to. The server responds by sending a denied access response

The following diagram is in SOA 3.0 methodology

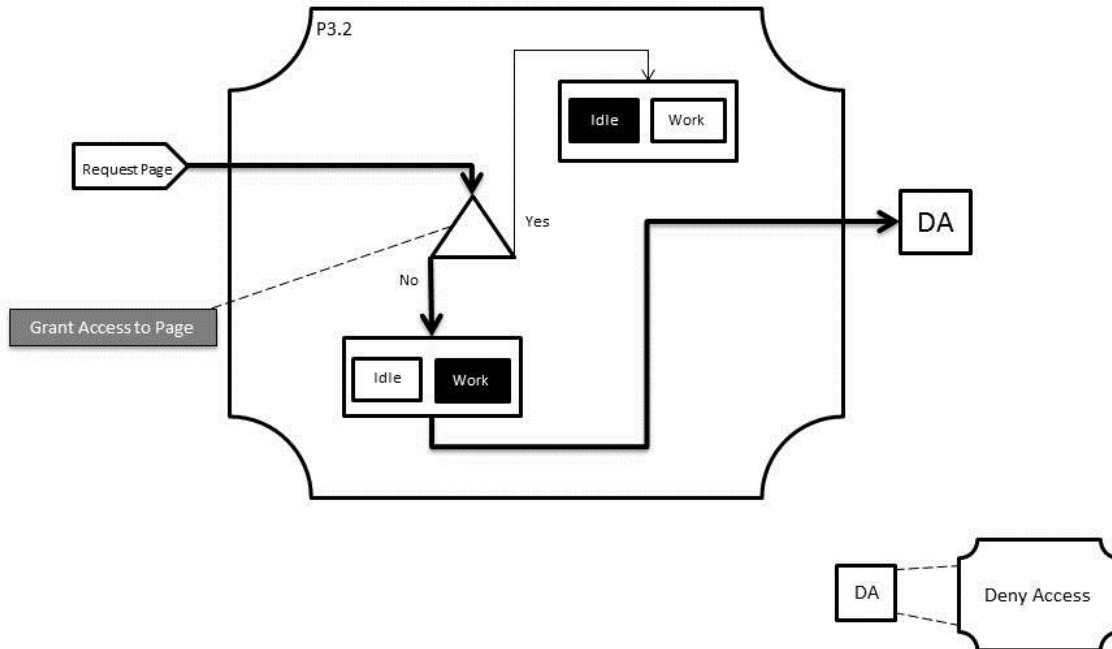


Figure 5.19: Page Request - Use Case B.

Logout - Use Case A: User sends log out request . The server responds by ending the session

The following diagram is in SOA 3.0 methodology

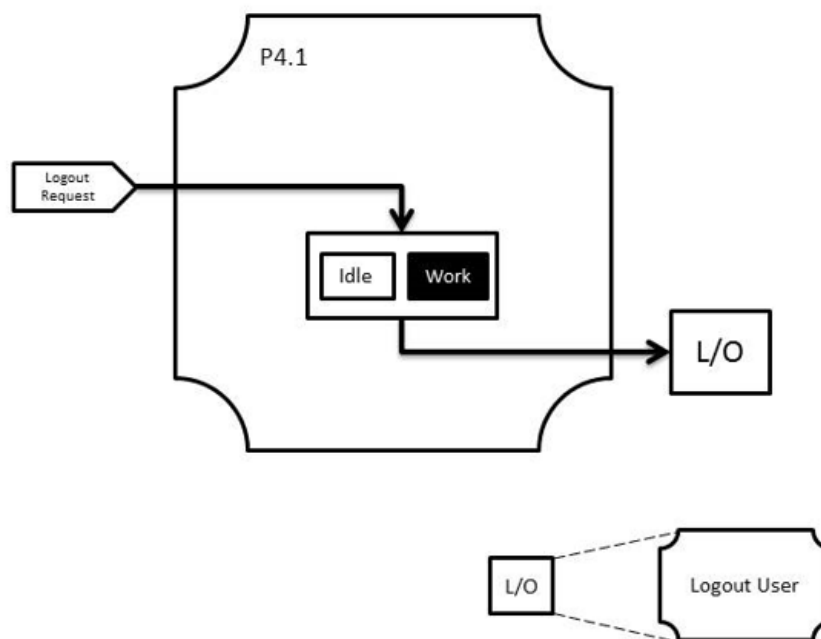


Figure 5.20: Logout - Use Case A.

5.4 Summary

Chapter 6 introduces a Master Observer Service (MOS) concept with an actual implementation and results with data bar charts related to Cloud Secured Gateway. To establish MOS, experimentation has been done and this experimentation is given the name MOS-Init. MOS-Init is a test service as shown in Figure 6.1 as **M**, which locates data from services requested by the consumer or a consumer service. As it is mentioned in Chapter 4, a service can be understood as a self-describing black-box like function, which can be called by another service or an application via its URL address that will provide a resultant dataset in response to a request by a service user or by another service.

6 Master Observer Service (MOS)

It is vital to conclude this thesis by providing the service observing framework in a Cloud and Governance associated to data security. We are living in the world of Cloud computing and services. The Cloud and users are being provided by the business world an increasingly diversified set of services to use in day to day activities to fulfill their requirements of buying, selling or other set of requirements using Cloud computing. In one or the other form, the services process and provide users information to use in the way they want to use it.

6.1 Introduction to MOS

The knowledge base is developed over a related set of information, and this information needs to be communicated among services and also needs to be stored in some data repositories for any future use. Almost all of the organizations use automated processes for their business activities. Due to the diversified business nature of several organizations, the gap exists of information transportation or message passing among several departments or partner organizations using Closed and Opensource technologies. This lack of proper information communication poses a challenge that needs to be resolved.

There are several service providers, who are now providing Cloud computing to the consumer organizations as mentioned in Chapter 2. A service is also proposed in this thesis, which

is called a Master Observer Service (MOS), and this service deals with the given tasks to help consumer organizations to effectively manage their users/customer oriented procedures. MOS makes data available for:

- Trouble-shooting of any malfunctioning servers
- Steering for available services to enumerate
- Facilitate services association in the Cloud

MOS provides a service URL addressing by determining a service's geographic latitude, longitude, and by inference, city, region and nation by comparing the URL's IP address with known locations of servers and load balancers in the Cloud. It is a fact that IP addresses can be dynamic as well as static. The URL assigned IP address can be reassigned to some other service's URL, which can be a relocation of this address within the same service provider or forwarded using some other mechanisms in an Enterprise Service Bus (ESB). As mentioned the job of data repository service is to store the searches of a requiring user or service, due to the large and increasing number of such stored data, it is not possible to be maintained in one repository due to the scalability of the data, this is where Hadoop helps by providing Hadoop Distributed File System (HDFS).

Initially MOS-Init was programmed and tested at University of North Dakota to be responsible for observing the selection and search of the required data by users. MOS-Init was recording the performance of data communication among several resultant dataset provider services. MOS-Init was configured as reactive data management, where the data repository service was informed about the status of a service, which was malfunctioning, it was the job of the service administering authority or service administrator to fix the malfunctioning service as receives the message generated by MOS-Init after storing the malfunctioning service's address and time it was

observed by the service administrator. This approach later was abandoned due to the increase in of the human intervention to manually manage the services. Several challenges have been found, which need more research. However, the challenges associated with this service are noted given below.

6.2 Challenges associated with Services' Performance

There are several challenges associated with the performance of services available in the Cloud; a few of those challenges are discussed below.

6.2.1 Availability of proper bandwidth: Cloud is a combination of several networks, running several operating systems and software as a service. Several data communication links have diverse capacity bandwidths that may differ by many orders of magnitude. It is also a possibility that one service is using the available bandwidth, which might be the lowest available bandwidth, this may be the service with the lowest static capacity or it may be a heavily loaded service from several requesting users or services with minimum available bandwidth. The service utilization changes as a function of time.

6.2.2 SLA issues among involved service providers: The messages communicated among services use the communication path, these message data packets are transported on a load balancer router as per the SLA among services or service providers and consumers. A service message paying more for the usage of a path is given an automatic priority and rest message packets can face delays.

6.2.3 Surge in Cloud access: Data packets or service messages in a Cloud transported within the services may vary extremely as a function of time. For example, there is typically many more services using Cloud data communication paths during peak work hours in contrast to nights and weekends. This is analogous to a weekday morning, such as Monday when there is a vehicle rush on the roads to a downtown as everyone is running to reach their office or workplace. Such increased surge in the Cloud can frequently introduce blockages in the message transportation among services.

Every service has an owner and some enforced policies to secure it and to work with. To look at the governance and security associated with ASOA solutions, are discussed with a suggested Messaging Security Service (MSS) as future work.

6.3 Service Governance

ASOA provides a combination of services to an organization in the shape of a building built using designed service blocks. These services follow some specific set of rules and regulations to process any requests. These rules are provided by the policy makers of all stakeholder organizations. Regulations are also an important element of the service processing according to the law of the country. It is a vital part to be followed as well to conduct proper business activities.

SOA governance deals with a central repository of services' information for any future reuse. It also deals with the rules defined by the policy makers and regulations of the country, where the service is being provided. A good example can be AMAZON.COM, which caters books

and other merchandise for USA, and AMAZON.CA serves Canadian customers. The governance also deals with the patterns of the use of these provided services and it also removes duplication of the interrelated messaging among one enterprise's wide use of services.

It is a most important task of any Service Oriented Architecture Team (SOAT) to make sure that they gather all relative information from business policy makers at the initiation of a service design. It does not matter that a service is designed for a unit of an organization or for several organizations. This important task is ASOA governance.

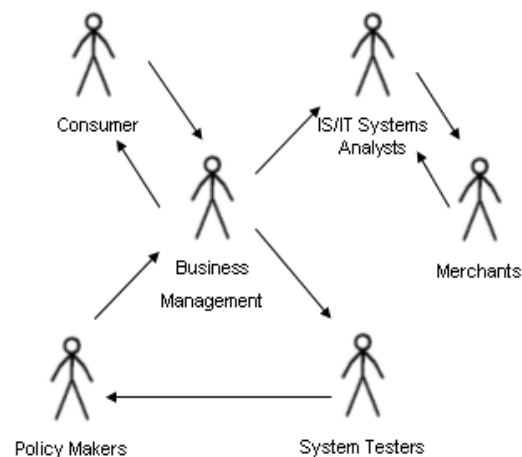


Figure 6.1: An ASOA Model of involved stakeholders

It is clear from the figure that there are several stakeholders in different business capacities involved in a SOA solution design. The uses of Demand For Alteration (DFA) by a selective team or Service Oriented Architecture Team (SOAT) in consideration of following key points can *get all* of the stakeholders a properly running service.

- Service design initiation should start after all related defined policies are studied by SOAT carefully with an agreement consent taken from all stakeholders.

- SOAT has to consider this agreement as tentative as the policy changes during or after service design can have an impact on service design or use.
- Services are to be delivered with an ownership to either one or a combined team of stakeholders to govern.
- Policies need to be embedded in service processing.

The governance of services needs to be secured. This point involves the security of the service(s) by using secured gateway algorithms presented in Chapter 4. The deployed service “A” need to be monitored and this monitoring can either be done by a management team or by another service “B” that has all rules and regulations in place in the processing phase of a consumer’s request processed by the deployed service “A”.

A successful ASOA service project can be considered as a success of all stakeholders. A service will be faster and more efficient, if it provides:

- Ease of use
 - User friendly interface
 - Clarity of request inputs
 - Acknowledgment of inputs
 - Error messaging
 - Input help availability
 - User feedback interface

- Information Hiding
- Policy based data processing
- Fault monitoring
 - Service security
 - User error detection
 - Software application error detection and logging
 - Hardware error detection and logging

Operational users of an organization or a consumer using these services are to be provided quality and secured services for their use and reuse.

6.4 Service Security

The services are designed to transfer important information in the form of messages from a Service **A** to System **X** or from a Service **A** to Service **B** and these messages are transferred usually on high security independent networks to other external networks using Cloud Computing Security Protocol as presented in this PhD thesis. This messaging between services can render the use of services to information leaks and intrusions by hackers or hacking agents, if is not protected by Cloud Monitoring Gateway. These hacking agents are also in the form of services, these are also known as Man in the Middle, to get to certain information, which might be used in an unwanted activity for the organizations or against the law.

This is the main reason that the policies describing security rules for the service control should be very strict in general. A security service can be introduced and embedded in all ASOA solutions to provide a pre-approval for all messaging among services as mentioned above. The jobs of this Messaging Security Service (MSS) can be as follows:

- MSS is to establish the protection of messaging, as various services might be directly connected to external networks.
- MSS should be able to clearly differentiate among authorized messages for external transfer from those messages, which cannot be allowed due to a predefined non-shareable security policy.
- MSS should be able to use an encryption technology to prevent the creation of false messages.

SOAT has to select a secure operating system that safeguards from external intrusions. This operating system should consist of service generated messaging transfer with security process management functions. It is the job of policy makers to work with SOAT to establish rules to validate and classify internal operational users and service monitors. This classification of operational users and service monitors is a prerequisite to allocate responsibilities for user actions, in handling day to day service logging and managing error reports to get corrected by IS/IT departments.

6.5 ASOA MSS Model

ASOA as mentioned earlier, proposes a Messaging Security Service (MSS) model using Cloud Monitoring Gateway, which is pictorially represented in the given Figure 6.2. This model contains a life cycle of MSS with a design time involvement of human factor.

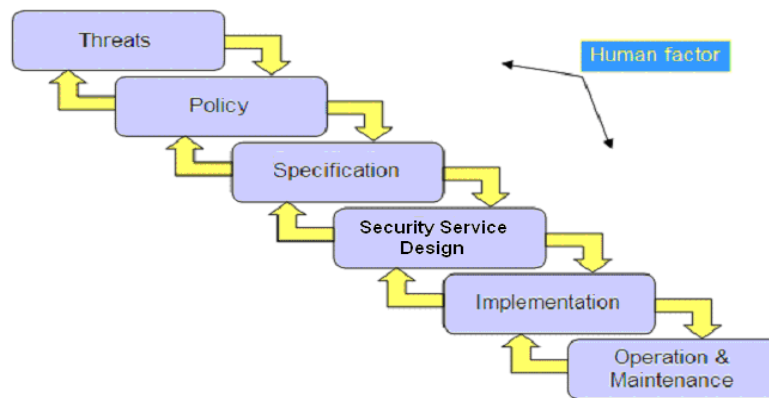


Figure 6.2: ASOA Security Service Design Life Cycle

The threats can be both external and internal. Policies are to be discussed by SOAT and policy makers in relation to the services is one of the vital tasks to be completed during the design time of the service to establish an MSS.

- **Threats:** Few threats can be classified as follows:
 - **Denial of Service or DoS** [68, 69] is the unavailability of services. The systems responding to such attacks can also be crashed. DoS can also be understood as a flood of requests to a service with more requests than the service can manage to serve.

- **Dictionary Attacks** [70] are a common way to crack a service to gain an unauthorized access. Attackers constantly use common usernames and password combinations to gain access of the service by weak username and password combinations.
- **Code injection** [71] attacks are comparatively uncomplicated and generally involve some acquaintance of technology the back-end system is using behind the service interface. A malicious code is injected by a service hacker to take the service control, by changing its security parameters, some hackers also tend to disclose these parameters for other attackers. As a result, several hackers and attackers either hijack the service or monitor it.
- **Policy** is a vital part of MSS. It is a very well understood fact that the dependence on and increase in the use of SOA in critical IT/IS projects of our industry need an inclusive security policy developed. A service security violation can cause serious legal, non-lucrative issues damaging the perception of an organization. The development of SOA solution's security is suggested by ASOA as a primary consideration when establishing communications between all involved stakeholders and Cloud service providers. Services Security Policy [72] is defined as a meticulous set of base contentions unfolding the privacy and reliability guarantees supplied by encryptions and signatures established within a message.
- **Specification** phase of an MSS model contains all the information related to probable anticipated threats and a completed security policy combination provided by all stakeholders, which leads to an infallible security service design.

- **Security Service Design** is a model of a precisely planned security approach of the SOA solution that puts its focus on the three fundamental principles given below:
 - Confidentiality
 - Integrity
 - Availability

- Once the modeling of a security service is complete, it needs to be implemented and this implementation needs a continuous revisit of Service Security Admin to add any new threat levels, if found during the study of the feedback of use of services. This is also a prime job of security admin to operate and maintain the security policy as being cautious to any probable inner as well as outer attacks that might introduce Denial of Service (DoS).

6.6 Analysis and Results of MOS-Init Usage

An analysis is presented here, which shows the calculation of the probable service failures if not protected by Cloud Monitoring Gateway, attacks by the hackers on services and the success of the deployed services using Poisson distribution.

6.6.1 Services Failure Prediction: The given calculation is for the probable attacks by hackers on services or the failure of some essential element of the service. This can be a key data transmission mode, such as a router or failure of a data storage server of web services used in a Cloud using Poisson distribution. This type of distribution is suited to find the attacks or service

failures as occurrences by some hackers or by some unknown causes over a specified time interval.

Let us see the Poisson distribution notations:

“ x ” (Random variable) = number of attacks in an interval

Or

“ x ” (Random variable) = number of service elements failures in an interval

The Poisson distribution function is defined as follows:

$$f(x) = e^{-\lambda} \lambda^x / x!$$

There are few elements associated with this distribution function, λ is the mean and the standard deviation of this mean is also associated.

$$\text{Mean} = \lambda$$

We will be using Poisson distribution to approximate the binomial distribution in the given set of rules for attacks or failures as described in the start of this section.

$n > 100$ n is number of occurrences of attacks or failures

$np < 10$ The probability of success of these occurrences is np

We will be using Poisson distribution with $\lambda = np$. These tests are performed on the data provided by an ABC Cloud service provider or ABC-CSP, which reports that over a 20 month time period, 196 service failures in total occurred of data communication and data server shutdowns happened, we can find the mean number of service failures per year:

$$\lambda = 196/20 = 9.8 \text{ Failures happened per month}$$

Now we will try to find the probability, how many service failures might be probable next year for our service provider?

$$f(x) = (9.8)^x \cdot e^{-9.8} / x!$$

In case there are no failures next year, we will put $x = 0$.

$$f(0) = (9.8)^0 \cdot e^{-9.8} / 0! = 0.0000555$$

Let us find out, if there are 5 failures are predicted next month.

$$f(5) = (9.8)^5 \cdot e^{-9.8} / 5!$$

$$f(5) = 0.042$$

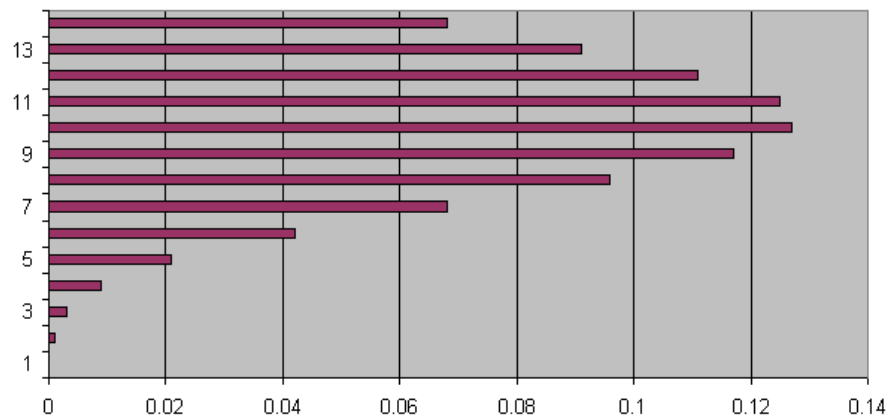


Chart 1: The expected failure of next month

Let us consider another example for the given Cloud Service Provider; Poisson distribution is used to find the probable number of attacks on that ABC-CSP in the year. We are going to work on the given conditional hypothesis:

- Provision 1 seems practically rational; that the chance of the CSP being attacked seems likely to be about the same for each day during the year.
- Provision 2 also seems practical; that an attack is independent of another attack.
- Provision 3 is in all satisfactory probability, unless these hackers are prone to go on rampages and launch all sorts of attacks, such as DoS, and Dictionary Attack at once.

Thus it is expected that the Poisson distribution can be used for this type of data modeling. The data in Table **A** are based on the 280 services and weeks combination (14 services in each of the 20 weeks of the study). For the ease of understanding we will deem these services to be 280 different services, each observed for one week. For each service, the number of attacks within the week was recorded. As we have the data available of these 280 different services, an empirical analysis can be done of what the probabilities are for 0 attacks per service, 1 attack per service, and this list goes on.

$$f(x) = e^{-\lambda} \lambda^x / x! \quad \text{where } \lambda = 14/20 = 0.71$$

On average, only 0.71 attacks per service in this manner per week were found. By applying Poisson distribution the following data have been generated for a graphical representation to understand the trend in Table **B** of attack on the services.

Attacks	Mean			
	0.1	0.7	1	5
0	0.9048	0.4966	0.3679	0.0067
1	0.0905	0.3476	0.3679	0.0337
2	0.0045	0.1217	0.1839	0.0842
3	0.0002	0.0284	0.0613	0.1404
4	0	0.0050	0.0153	0.1755
5	0	0.0007	0.0031	0.1755
6	0	0.0001	0.0005	0.1462
7	0	0	0.0001	0.1044
8	0	0	0	0.0653
9	0	0	0	0.0363
10	0	0	0	0.0181
11	0	0	0	0.082
12	0	0	0	0.0034
13	0	0	0	0.0013
14	0	0	0	0.0005
15	0	0	0	0.0002
16	0	0	0	0

Table 6.1: Attack on the Services

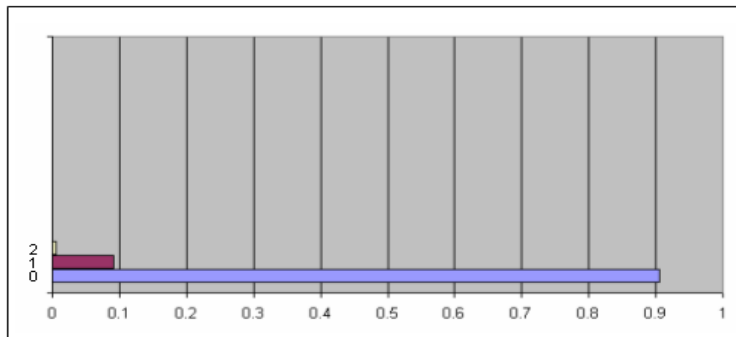


Chart 2: Attacks depiction on mean is 0.1

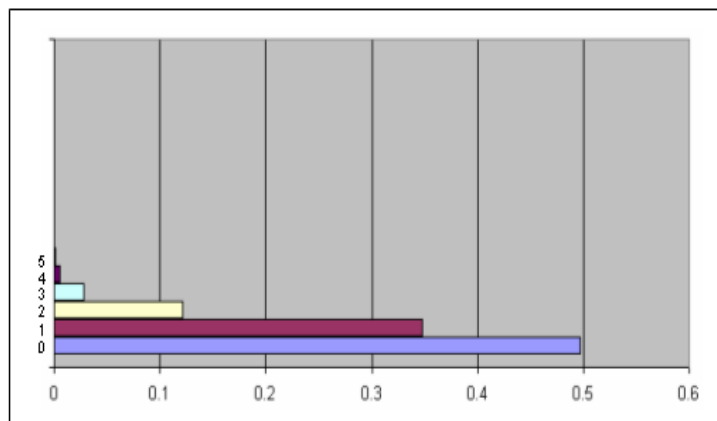


Chart 3: Attacks depiction on mean is 0.7

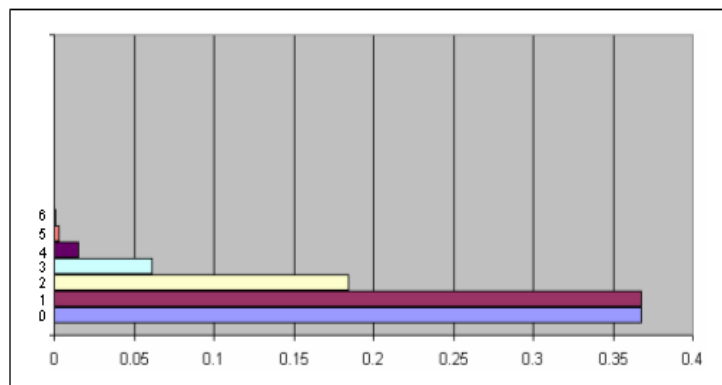


Chart 4: Attacks depiction on mean is 1.0

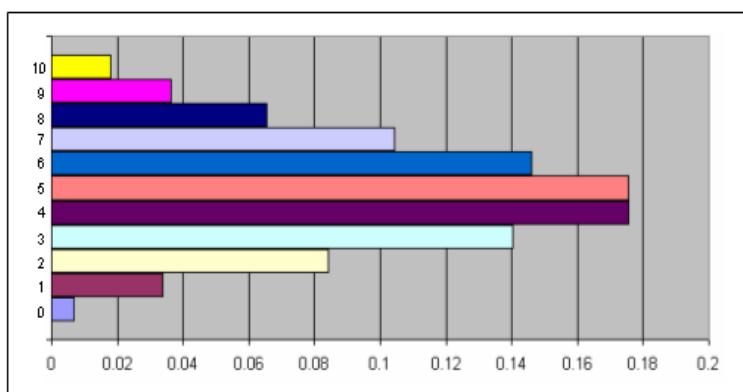


Chart 5: Attacks depiction on mean is 5.0

6.6.2 Success of Services Utilization: Let us consider another hypothetical example; in a Cloud datacenter two services are being used every hour on last Monday. What is the possibility that exactly three services will be used at the exact same hour on next Monday?

$$e^{-\lambda} = ?, \text{ where } \lambda = 2, e = 2.718$$

$$e^{-\lambda} = (2.718)^{-200}$$

$$e^{-\lambda} = 0.135$$

$$\lambda^x = ?, \text{ where } \lambda = 2, x = 3$$

$$\lambda^x = 2^3 = 8$$

$$f(x) = ?,$$

$$f(x) = e^{-\lambda} \lambda^x / x!,$$

$$f(3) = (0.135)(8) / 3!$$

$$f(3) = 0.18$$

The use of three services on the same hour on next Monday has 18% possibilities.

6.7 Summary

There are several methodologies currently being utilized to design web services, such as UML, BEPL etc., and the use of any of these needs a good knowledge of the notations associated to these approaches. This thesis presents an uncomplicated software engineering methodology, called ASOA using System's Requirements Design (SRD) methodology. It is a complete system

engineering, architecting, and system lifecycle modeling approach that can be used by the software engineering community to achieve the desired results for any stakeholders' desired web/Cloud service(s) expeditiously and as accurately as possible. The research work done in this chapter also provides another framework to observe a successful service. The issues related to service governance and security are given to clarify the need to resolve during a service design phase.

7 Conclusion & Future Work

As described in this PhD thesis, though there are advantages in using Cloud-based systems, there are yet many realistic problems which have to be solved like Big Data analytics. This type of analytics needs a faster analysis on the data, which is moving with high velocity. This research sheds light on a few of the several issues dealing with Cloud and specifically the end-user sessions that need to be generated for any analytics using Hadoop over MapReduce.

7.1 Conclusion

Given the ubiquity of wired and wireless high speed connections, a service in a Cloud is a phenomenon of seamlessly transmitting secured data between n nodes after making sure that the user is authenticated to work on their related sessions, which might or might not include Big Data. Intermittent connectivity of the Cloud is a main reason to use a good mix of the approaches to transmit data asynchronously. Cloud is a union of several hardware and software platforms. There are several hybrid technologies being utilized to provide the services in the Cloud by many service providers. This research sheds light on a few of the several issues dealing with Cloud and specifically the security issues and research work done to find suited solutions to resolve security problems in major. It also encompasses a proposal of Cloud Security Architecture based on Layered Architecture approach. This approach is presented as Cloud Security Protocol (CSP), with

expansion of three services with an addition of encrypted internal keys to find the source of the request and the destination, which needs more future enhancements and modifications with the incorporation of systems' functional testing. CSP has potential to be enhanced to achieve a favorable methodology to resolve the issues related to Cloud security and reliable availability of the desired results from the effective use of Cloud's available resources. Since, Big Data is now a reality that we are dealing with, it is vital to make sure that our work sessions involving Big Data are secured, as well.

7.2 Realization

Big Data Analytics in a secured way by using Data Science algorithms can give an organization the uniqueness of the ideas, which they can embed in their decision support systems. It is a known fact that an idea is the beginning of anything, such as a horseless carriage, which we call an automobile, a flying machine also known as airplane, etc. If an organization is not able to define the unique value, by not utilizing either Role Based Access Control (RBAC) or any Security Protocol that they can embed to make their customers feel secured, this will make them the least competitive. Big Data Analytics using both Data Science and Cloud Computing is the key to establishing this niche of uniqueness for the customers, whether you manufacture a pen, car, airplane, or you construct buildings or design furniture or even write books as an author.

Exponential growth of data in any way has brought us the boom of Big Data. Computers, Smartphones and tablets, sensors (RFID) in vehicles and other equipment, the use of social networks, online services by Cloud services providing organizations' applications (Cloud Computing / Software as a Service, Platform as a Service, Infrastructure as a Service) and the rest

of the Web in terms of news media, blogs generate alone an immense amount of data. Hadoop, YARN (Yet Another Source Negotiator), and Spark introduced improved techniques of data Storage, Processing & Transformation (SPT). ETL (Extract, Transform & Load) is a common term, however the introduction of SPT is an evolution to ETL. This PhD dissertation is also presenting this novel term of SPT for future researchers to conduct research and experimental studies.

The decisions based on SPT (Storage, Processing & Transformation) for Big Data solutions can help companies to make better decisions, for example, market potential and customer behavior to better estimate a future course of action to reach customers on a better note in terms of products, services, etc. The service is the key here, no matter whatever the product an organization has, it might be the product from heaven, if the service provided to a consumer is lousy, then the organization will not get a customer-base. In the research and development of new solutions we can find by means of simulations and experiments using Data Analytic techniques to improve service or optimize the serving ways. The logistics and distribution of goods by an entity to help the flood of data to optimize operations and processes can result in a reduction in costs. Controlling Big Data can help in fraud prevention and detection of other business risks, this will also boost the trust of the customer in the organization.

We must look at the privacy requirements of our customers, this needs to be considered early on in Big Data projects, in the design phase and the use of SPT techniques can optimize Big Data processing for better decision making. We can achieve the Privacy Compliance as part of our root design of such applications/reports. This can be ensured without incurring any later legal risks, time and cost-intensive project adjustments are to be considered as a necessary aspect. Use of ASOA can elaborate SPT in depth.

7.3 Limitations

This thesis sheds light on how an organization can step into the era of Big Data and Cloud, by having right strategy using provided Security Gateway algorithms. Any strategy, which is good in nature and well thought out, while in the phase of designing, cannot succeed, if poorly executed. It is vital that the organizational leadership has a sense of how to execute the strategy. The very beginning for any organization is to figure out collectively within, what are the strategic Big Data Security challenges that they can face, and which challenge is the biggest. In case they do not implement Big Data Analytics Secured sessions, in other words ASOA strategy in this thesis, they will need to determine what factors can critically impact the organization, in terms of their marketplace competitiveness.

Strategy is not a goal in terms of Closed or Opensource technology utilization. In case some manager says that their organizational strategy to use either/or, is to be the number one company in the marketplace or their strategy is to grow their services level of satisfaction faster than their competitors, this is absolutely a wrong approach. These are goals and aspirations, these are not strategies. As we all know, money plays an integral role in any of our industries, whether, it is retail, travel, healthcare or academics. In the era of Big Data, we have seen Opensource software/tools playing a huge role and are available to public to learn at almost zero cost.

Big Data can be considered a decision making factor, thus inducing a wrong approach or algorithm to generate reduced results, at the time we use either Hive, Pig or Impala or MapReduce from the direction, which is not focused on certain optimized results can derail the prospects offered by these tools and techniques.

Let us have a look at some analytics in the financial sector as an example here. This example is in actuality a study which can be primarily used to create a holistic view for the stakeholders, searching for the golden needle from the hay stack. This working example allows a more personalized approach to the factors influencing the choice of the target audience for a business to focus on to uplift their service or products delivery. Rather on a simpler note, by improving communication and other sales and marketing strategies, so that they can improve their organizational performance to generate a better revenue stream.

This example takes a bank serving on a global basis, let us say it as ABC Bank. This institution uses its operational approach using Big Data to improve services provision to the clients and enhance reputation among multinational fields to generate goodwill, while earning better revenues. This study shows the use of customer service provided directly to the clients using e-mails between customers and customer service. These customers have some payments to be given back to the bank and are trying to avoid either paying interest or buying in more time to create delays in their scheduled payments. This is a risk that every bank has to take while issuing credit cards to its clients, no matter if it is an individual or a business (small, medium or big) to the degree of such risk and/or different sociological factors in order to establish the best means of recovery of the debts that these clients get into.

A set of algorithms can be written to help this bank by running within streams of Big Data in a secured fashion, as mentioned in this PhD thesis. This Bank has the data of the communication that has been conducted, as mentioned earlier, as well as the discussions over the phone (transcribed) and the extraction of emails, and text documents. This methodology will certainly help the decision makers of ABC Bank to estimate the state of confidence of customers to the bank or distrust of the bank to the (particular) client(s). This methodology is also helpful for any

organization, which is in the service industry of any type to assess the risk of fraud that can result in the loss of revenue and can create a distrust among the general population using such an institution.

The analysis conducted on Big Data in the Cloud using Security Gateway algorithm provided in this PhD thesis can help establishing secured usage for the organization, as well as providing insights of how the users are feeling towards both bowing to the organization and repaying their debts, or whether they are trying to find a run not to pay the organization. This type of analytics will also be helpful for the organization to recognize which of the employees are using influencing communicate to the client to make them either pay their debt or start paying in installments. This will allow the organization to recognize these employees and create training materials around these successful communication styles to improve the performance of the rest of the customer service providing employees to generate a harmony internally to bring most of the (about to be) lost credit, that can be helpful for future investments.

7.4 Future Work

Big Data is real and we all live in the world of evidence. We will be looking into Big Data Evidence theory for our future work, which is so far unheard of within the Big Data community. We have witnessed several advances in computational performance in terms of both traditional, and parallel programming aspects in terms of Java, Python, C++, and more, whereas the world of Big Data has reintroduced Functional Programming, such as Scala in major, which have brought us the design and development of high-performance computing simulation tools using Hadoop, YARN, Storm and Spark.

It is a fact that we have to account for uncertainty even in Big Data streams, when corrupted data can stream as well, while generating such high-performance systems using either MapReduce or YARN within Hadoop Appliances can fail in service performance predictions. This is also known as Predictive Analytics. Traditionally we have seen that evidence theory has been utilized to measure uncertainty in terms of the uncertain measures of belief and plausibility on the basis of whatever the data we have. It is also witnessed in the computing community that Cloud computing has provided a flexible and scalable infrastructure to grow beyond contemporary borders to the organizations as well as the users' everyday use of services. Both Big Data and standard Storage Area Networks (SAN) provide data storage and ETL using Cloud.

Computing in any form is a very structured way or a set of instructions to resolve our everyday issues, whether it is academic or industrial. It is an undoubted fact that uncertainty is everywhere. This is one of the reasons that there has been an augmented prominence focused on accounting for the diverse forms of uncertainties that are initiated in the mathematical models and simulation tools. A mixture of forms of uncertainties exists and it is significant that each one of them is accounted for by proper means depending upon the amount of available information. Big Data is also not free of uncertainty.

We can distinguish three different types of uncertainty, and we know them as epistemic uncertainty, aleatory uncertainty and error. The inherent variation of the physical system can be described by the term aleatory uncertainty. Such discrepancy can occur due to the unsystematic nature of the input data that is streaming for a retailer or from a social media to an analytic Data Science analytics provider, and can be mathematically represented by a probability distribution once enough experimental data is available. Epistemic uncertainty in non-deterministic systems happens due to ignorance, lack of knowledge or shortened information. Error is considered to be

dissimilar from uncertainty though several researchers submit to error as “numerical uncertainty.” Error is defined as an identifiable deficiency in any stage or movement of modeling and simulation that is not due to lack of knowledge. The challenges of the present technological climate can be resolved by software engineers and decision makers, such as CTOs or MIS managers, by aligning business needs, using service components to improve service to consumers. The marriage of SOA and Big Data architecture is a boon to an enterprise looking to create service components in an agile fashion and reuse existing components’ infrastructure.

References

- [1] Dean, Jeffrey. Sanjay Ghemawat.: MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM 51.1, Pages 107-113, 2008
- [2] Atif Farid Mohammad, Hamid Mcheick, Emanuel S. Grant (chapter). Cloud as a Major Catalyst of Change in Contemporary Business Environment, published by Cambridge Scholars Publishing, Book title: Business Intelligence and Mobile Technology Research: An Information Engineering Perspective, Edited by Sean B. Eom & Mohamad Laouar, New Castle Tyne, United Kingdom, 2014.
- [3] R. Schreiner, U. Lang; Protection of complex distributed systems. Proceedings of the 2008 workshop on Middleware security. Pages 7-12. 2008.
- [4] T. Erl. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, 2005.
- [5] <http://www.amazon.ca/>; Accessed on April 03, 2013.
- [6] <http://www.amexcorporatipayments.ca/ResourceLibrary/WhitePapers.aspx>; Accessed on October 13, 2014.
- [7] A. Korostelev, J. Lukkien, J. Nesvadba, Y. Qian, “QoS management in distributed service oriented systems”, Proceedings of the 25th IASTED International Multi-Conference, pp. 345-352. 2007
- [8] E. Borcoci1, A. Asgari, N. Butler, T. Ahmed, A.Mehaoua, G. Kourmentzas, S. Eccles, “Service Management for End-to-End QoS Multimedia Content Delivery in Heterogeneous Environment”, Proceeding of Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/ E-Learning on Telecommunications Workshop, pp. 46- 52. 2005
- [9] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J.Kalagnanam, H. Chang, “QoS-Aware Middleware for Web Services Composition”, IEEE Transactions On Software Engineering, Vol. 30, No. 5, pp. 311. 2004
- [10] H. Huynh, E. Lavinal, N. Simoni, “A Dynamic QoS Management for Next Generation of Services”, Third International Conference on Autonomic and Autonomous Systems, pp. 11-11. 2007
- [11] X. Gu, K. Nahrstedt, “QoS-Aware Service Composition for Large-Scale Peer-to-Peer Systems”, Book Chapter: Grid Resource Management: State of the Art and Future Trends, Chapter 24, pp. 395 - 410, Kluwer Academic Press, October 2003
- [12] W. Dostal *et al.*: Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis. Heidelberg, Elsevier, 2005
- [13] F. Kupsch, and Werth, D.: Process-Driven Business Integration Using Peer-to-Peer Technology. In: M. Khosrow Pour (Ed.): Innovations Through Information Technology, pp. 1294-1300. 2004
- [14] Hengheng Xie, Azzedine Boukerche, Ming Zhang, Bernard P. Zeigler. Design of A QoS-Aware Service Composition and Management System in Peer-to-Peer Network Aided by DEVS. in Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications, pp.285-291. October 2008
- [15] Christian Dörner, Volkmar Pipek, Moritz Weber, Volker Wulf. End-user development: new challenges for service oriented architectures. in Proceedings of the 4th international workshop on End-user software engineering, pp. 71-75. May 2008

- [16] Brooks, F.P.J. No silver bullet: essence and accidents of software engineering, IEEE Press, pp.10-19. 1987
- [17] Atif Mohammad, Hamid Mcheick, and Emanuel Grant. Use of SOA 3.0 in Private Cloud Security Gateway Service Design: Beginning of New Era, submitted to IEEE transactions, special issues Cloud Computing, 2014
- [18] Atif Farid Mohammad, Emanuel Grant, Ronald Marsh, Scott Kerlin. Cloud Computing Monitoring Gateway for Secured Session Management of Big Data Analytic Sessions. CGAT 2013, Singapore April 2013
- [19] Atif Farid Mohammad and Hamid Mchick. Cloud Services Testing: An Understanding. 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011), September 2011
- [20] Atif Farid Mohammad, Emanuel Grant, Scott Kerlin. Cloud Computing Monitoring Gateway for Big Data Secured Analysis using Live Signature – TPALM. CGAT 2013, Singapore April 2013
- [21] Atif Farid Mohammad, Assion Lawson-Body, Emanuel S. Grant. SOA and Security of Loosely Coupled Services in the Cloud using SOA 3.0. Software Engineering Research and Practice 2010: 481-487
- [22] A Beloglazov, J Abawajy, R Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Future Generation Computer Systems 28 (5), 755-768 2012
- [23] <http://www-3.4.ibm.com/shop/americas/webapp/wcs/stores/servlet/default-ProductDisplay?productId=4611686018425790402&storeId=124&langId=124&categoryId=4611686018425185805&dualCurrId=75&catalogId=-124>; Accessed on Jan 31, 2012
- [24] http://configure.us.dell.com/dellstore/config.aspx?oc=becwtk1&c=us&l=en&s=bsd&cs=04&kc=productdetails~pedge_2970_rack; Accessed on Jan 31, 2012
- [25] <http://h10010.www1.hp.com/wwpc/us/en/sm/WF06b/15351-15351-3328412-241644-241475-3579703-3633169-3633172.html>; Accessed on January 31, 2012
- [26] <http://www.mendax.com/serverInfo.aspx?productID=543883>; Accessed on Jan 31, 2012
- [27] C. Kopparapu. Load Balancing Servers, Firewalls, and Caches. John Wisely & Sons Inc., 2002
- [28] <http://www.gartner.com/it/page.jsp?id=707508>; Accessed on January 31, 2009
- [29] <http://aws.amazon.com>; Accessed on January 31, 2009
- [30] <http://www.emc.com/products/category/subcategory/cloud-optimized-storage.htm>; Accessed on January 31, 2012
- [31] <http://www.apтана.com/cloud?video>; Accessed on February 10, 2012
- [32] <http://www.gogrid.com>; Accessed on February 10, 2012
- [33] docs.google.com; Accessed on February 10, 2012
- [34] Greg Goth, "Software-as-a-Service: The Spark That Will Change Software Engineering?" IEEE Distributed Systems Online, vol. 9, no. 7, art. no. 0807-o7003, 2008
- [35] Dan Ma. The Business Model of "Software-As-AService", IEEE International Conference on Services Computing 2007
- [36] Sathyan, J.; Shenoy, K. Realizing unified service experience with SaaS on SOA. in Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference, Page(s):327 - 332, Jan. 2008

- [37] Hammer, Michael. & Champy, James. Reengineering the Corporation. HarperCollins Publishers, Inc., New York, NY 10022. 1993.
- [38] Abdolvand, Neda. Albadvi, Amir. & Ferdowsi, Zahra. Assessing Readiness for Business Process Reengineering. IT Department, Engineering Faculty, Tarbiat Modares University, Tehran, Iran. P. 497-510. 2007
- [39] M. Rosen, B. Lublinsky, K.T. Smith, M. J. Balcer. Pulished. Applied SOA: Service-Oriented Architecture Design and Strategies. Wiley Publishing Ltd. 2008.
- [40] Herzog, N. V., Polajnar, Adrej, & Tonchia, Stefano. Development and Validation of Business Process Reengineering (BPR) Variables: A Survey Research in Slovenian Companies. International Journal of Production Research. Vol. 45, No. 24, 5811-5834.
- [41] Newman, Mike. & Zhao, Yu. The process of Enterprise Resource Planning Implementation and Business Process Reengineering: Tales from Two Chinese Small and Medium-Sized Enterprises. Info Systems J 18, 405-426. 2008
- [42] Atif Farid Mohammad; The path from a Legacy System to GUI System. IEEE Conference on Systems, Computing Sciences and Software Engineering. Bridgeport, CT. December 2008.
- [43] Peter H. Feiler. Reengineering: An engineering problem. Technical Report CMU/SEI-93-SR-5, Software Engineering Institute, Carnegie Mellon University, July 1993
- [44] Jean-Marc DeBaud, Bijith M. Moopen, and Spencer Rugaber. Domain analysis and reverse engineering. In Proceedings of the International Conference on Software Main-tenance (Victoria, BC, Canada; September 19-23, 1994), pages 326-335, 1994.
- [45] Atif Farid. Mohammad, Hamid. Mcheick. Cloud as a Major Catalysts of Change in Contemporary Business Environment. ICSSENT 2012, LNCS Springer Publications. Tunisia, December 2012
- [46] Atif Mohammad and Hamid Mcheick, A Pre-Cloud Evolution Perspective of Legacy Systems, Proceedings of 2013 International Conference on Information Technology and Electronic Commerce (ICITEC-IEEE 2013), Harbin, China, 2013
- [47] H. Karhunen, M. Jäntti, and A. Eerola. Service-Oriented Software Engineering (SOSE) Framework. In Proceedings of 2005 International Conference on Services Systems and Services Management (ICSSSM). IEEE, 2005
- [48] F.-R. Lin, M.-C. Yang, and Y.-H. Pai. A generic structure for Business Process Modeling. Business Process Management Journal, 8(1):19–41, 2002
- [49] B. Korherr and B. List. An Evaluation of Conceptual Business Process Modelling Languages, in 21st ACM Symposium on Applied Computing (SAC'06). ACM Press, 2006
- [50] A.-W. Scheer. ARIS - Business Process Modeling. Springer, 2000
- [51] R. Mayer, C. Menzel, M. Painter, P. deWitte, T. Blinn, and B. Perakath. Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report. Technical report, Sept. 1995
- [52] Object Management Group (OMG). Business Process Modeling Notation, Specification, Version 1.0. 2006
- [54] H. Kim. Conceptual Modeling and Specification Generation for B2B Business Processes based on ebXML. SIGMOD Rec., 31(1):137–145, 2002
- [55] G. Kramler, E. Kapsammer, G. Kappel, and W. Retschitzegger. Towards Using UML 2 for Modelling Web Service Collaboration Protocols. In Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA'05), Feb. 2005

- [56] OASIS Web Services Business Process Execution Language (WSBPEL) TC. Web Services Business Process Execution Language Version 2.0, Jan. Version 2.0, Committee Specification. 2007
- [57] F. Leymann, D. Roller, and M. T. Schmidt. Web services and business process management. *IBM Systems Journal - New Developments in Web Services and E-commerce*, 41(2), 2002
- [58] UN/CEFACT. UN/CEFACT - ebXML Business Process Specification Schema, Version 1.11. Nov. 2003
- [59] Object Management Group (OMG). Business Process Modeling Notation, Specification, Version 1.0. 2006
- [60] Atif Farid Mohammad; The path from a Legacy System to GUI System. IEEE Conference on Systems, Computing Sciences and Software Engineering. Bridgeport, CT. December 2008
- [61] Atif Farid Mohammad, Dustin E. R. Freeman; Supply Chain Requirements Engineering: A Simulated Reality Check. IEEE Conference on Systems, Computing Sciences and Software Engineering. Bridgeport, CT. December 2008
- [62] C. Koppurapu. Load Balancing Servers, Firewalls, and Caches. John Wisely & Sons Inc., 2002
- [63] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI'04, 6th Symposium on Operating Systems Design and Implementation, Sponsored by USENIX, in cooperation with ACM SIGOPS, pages 137–150, 2004
- [64] <http://research.google.com/archive/mapreduce.html>: Accessed on February 09, 2014
- [65] Atif Farid Mohammad, Emanuel Grant. Cloud Computing Security for Governments using Live Signature TPALM Printing Client Service. *Journal of Computing* Vol 2, No. 3. 2012
- [66] Atif Farid Mohammad, Emanuel S. Grant., Cloud Computing Security Gateway Proposed Service Architecture. International Conference on Cloud Computing and Visualization 2012 (CCV2012), Bali, Indonesia, April 2012.
- [68] Cert. Incident note IN-2004-01 W32/Novarg.A virus. 2004
- [69] K. POULSEN. FBI busts alleged ddos mafia. www.securityfocus.com/news/9411. 2004
- [70] R. Canetti, S. Halevi, and M. Steiner. Mitigating dictionary attacks on password-protected local storage. In CRYPTO, pages 160–179, 2006
- [71] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of ipv6 packets over ieee 802.15.4 networks (rfc 4944). Technical report, IETF, September 2007
- [72] G. Della-Libera, P. Hallam-Baker, M. Hondo, T. Janczuk, C. Kaler, H. Maruyama, N. Nagarathnam, A. Nash, R. Philpott, H. Prafullchandra, J. Shewchuk, E. Waingold, and R. Zolfonoon. Web services security policy language (WS-SecurityPolicy). Version 1.0. 2002
- [73] Pierangela Samarati, Sabrina de Capitani di Vimercati, “Access Control: Policies, Models, and Mechanisms”, *lecture Notes in Computer Science*, Vol.2171, 2001, Pages 137
- [74] Alqahtani, S.M.; Gamble, R.; Ray, I. "Auditing Requirements for Implementing the Chinese Wall Model in the Service Cloud", *Services (SERVICES)*, 2013 IEEE Ninth World Congress on, On page(s): 298 - 305
- [75] John A. Miller, Mei Fan, Shengli Wu, Ismailcem B. Arpinar, Amit P. Sheth, Krys J. Kochut, “Security for the METEOR Workflow Management System”, Large Scale Distributed Information Systems Lab (LSDIS), Department of Computer Science, the University of Georgia, <http://LSDIS.cs.uga.edu>

- [76] Patrick Brézillon¹ and Ghita Kouadri Mostéfaoui, “Context-Based Security Policies: A New Modeling Approach”, Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW’04), IEEE, 2004, pages 154 Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International, vol. 1, 2004, 72-77.
- [77] R. K. Thomas, R. S. Sandhu, “Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management”, Proceedings of the IFIP WG11.3 Workshop on Database Security, Lake Tahoe, California, August 11-13, 1997
- [78] William Tolone, Gail-Joon Ahn, Tanusree Pai, Seng-Phil Hong, “Access Control in Collaborative Systems”, ACM Computing Surveys, Vol. 37, No. 1, March 2005, pp. 29–41.
- [79] Zettaset, The Big Data Security Gap : Protecting the Hadoop Cluster. (n.d.),2014
- [80] J. Singh, “Big Data : Tools and Technologies in Big Data,” vol. 112, no. 15, pp. 6 – 10, 2015

Addendum

The Cloud Secured Gateway Service presented in this thesis receives requests to generate secured sessions for the user to work/process their data, namely for ETL. A Block ID is used to prevent a threat from accessing the server. The database of blocked users utilises a time sheet schema, which ranges from x min to the permanent blockades to potential threats. The database schema also has tuples that store such data like login and password. The login table schema individually stores the number of attempts the client tries to login and the use of the authorization key is also stored to generate sessions.

The Sessions table stores the log-in access, key, and ID. The Block table stores all ID's that have been blocked with locations with timer. Checking the ID at the gateway service means that the server will find the session ID and verify that it is the legitimate user, and also will verify that the ID is in a “safe network” as mentioned in chapter 4. It also makes sure that another user isn't using that same ID.

An idle state of Cloud Secured Gateway service is the state of waiting for a request from a client in the network. If the client requests to exit the session the server will record the ID and halt the session by deleting the session from active sessions. If the client requests to login it will check the ID, if the ID is valid, in this case it will insert a new session to the login sessions database. If the user is in a session the server will process the login attempt (Check_ID is in the login process). Finally, if the user is not of valid access, to the server, the server will record the ID of the Requested ID, and the request data. This will also notify the authorities about the threat attempt. The halt process will be executed with block ID storage. A tailorable Opensource is provided, which is implemented at a financial institution in March 2015.

Session Controller... Tailorable Opensource code

```

package com.openorg.appsecurity.controller;

import java.sql.Timestamp;
import java.util.Calendar;
import java.util.Map;
import java.util.Random;

import javax.servlet.http.HttpServletRequest;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.FileSystemXmlApplicationContext;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;

import com.openorg.appsecurity.dao.ActiveSessionDAO;
import com.openorg.appsecurity.dao.ActivityLogDAO;
import com.openorg.appsecurity.dao.CompletedSessionDAO;
import com.openorg.appsecurity.dao.BlackListIdDAO;
import com.openorg.appsecurity.dao.LoginSessionDAO;
import com.openorg.appsecurity.dao.SessionIDDAO;
import com.openorg.appsecurity.dao.UserDAO;
import com.openorg.appsecurity.dao.WhiteListIpDAO;
import com.openorg.appsecurity.model.ActiveSession;
import com.openorg.appsecurity.model.LoginSession;
import com.openorg.appsecurity.model.Session;
import com.openorg.appsecurity.model.User;
import com.openorg.appsecurity.utilities.AppProperties;
import com.openorg.appsecurity.utilities.ApplicationMailer;

/**
 * This class controls all interactions with the database sessions
 */
@Controller
public class SessionController {
    @Autowired
    private ActiveSessionDAO activeSessionDAO;

```

```

@Autowired
private CompletedSessionDAO completedSessionDAO;

@Autowired
private ActivityLogDAO activityLogDAO;

@Autowired
private LoginSessionDAO loginSessionDAO;

@Autowired
private UserDao userDao;

@Autowired
private WhiteListIpDAO whiteListIpDAO;

@Autowired
private BlackListIdDAO blackListIdDAO;

@Autowired
private SessionIdDAO sessionIdDAO;

@Autowired
private ApplicationMailer applicationMailer;

/**
 * This method controls the flow of all requests passed to the application. Using the
 * actionRequested
 * attribute in tandem with current conditions in the session tables, it determines the
 * correct course of action.
 * @param m The attributes passed from the AccessController
 * @return Data to be returned to the client
 */
@RequestMapping(value = "/SessionAction")
public @ResponseBody
String completeAction(Model m) {
    // Decrypt xc(Client_Request);

    //Retrieve data from the mapped attributes
    String username = (String) m.asMap().get("username");
    String password = (String) m.asMap().get("password");
    String authorizationKey = (String) m.asMap().get("authorizationKey");
    String ip = (String) m.asMap().get("ip");
    String actionRequested = (String) m.asMap().get("actionRequested");

    if (actionRequested.equals("logout")) {

```

```

        ActiveSession          active          =
activeSessionDAO.getSessionByUsername(username);
        if(active              !=              null      &&
authorizationKey.equals(active.getAuthorizationKey())){
            //Process the logout request
            processLogOutRequest(username);
        } else {
            //This session is no longer active, so the user should just be taken
back to the login page
            System.out.println("auth key incorrect");
            return "login";
        }
    } else if(actionRequested.equals("register")){
        //retrieve the phone number attribute
        String phoneNumber = (String) m.asMap().get("phoneNumber");
        try{
            //Add the new user to the registration table
            System.out.println("trying to add user to database: "+username);
            userDAO.saveUser(username, password, phoneNumber);
        } catch(Exception e){
            //Error is produced if user could not be added
            return "Error";
        }
        //Success if the user was added
        return "Success";
    } else if (loginSessionDAO.getSessionByUsername(username) != null) {
        //This is the second step of the login process. The user is currently in a login
session
        //and needs to be authenticated.
        return processLoginRequest(username, password, ip, authorizationKey);
    } else if (actionRequested.equals("login")) {
        //Retrieve user from database
        User user = userDAO.getUserByUsername(username);

        //Check if the user's ip is whitelisted, and if it isn't give them a chance to do
so.
        //This hasn't been implemented yet.
        if(user == null) {
            return "usernotfound";
        }

        if (!whiteListIpDAO.checkIfWhiteListed(ip)) {
            // give user chance to whitelist ip
            System.out.println("ip not whitelisted");
            String generatedKey = generateWhitelistKey();
            //Send a composed mail

```

```

        applicationMailer.sendMail("drew.mahrt@openorg.local", "Whitelist key",
"Here is your whitelist key: "+generatedKey);
        return "whitelist"+generatedKey;
    }

    //Check if the username+ip has been blacklisted
    if (!blackListIdDAO.checkIfBlackListed(username, ip)) {
        //Create a new session id entry in the sessionID table, and retrieve
it.
        int sessionId = sessionIdDAO.createId();

        //Create a new login session for the user
        loginSessionDAO.createSession(sessionId, username, ip);

        //Update the user table with the latest authentication key

        userDAO.updateAuthKey(username,loginSessionDAO.getSessionByUsername(u
sername).getAuthorizationKey());
        activityLogDAO.createLog(sessionId, new Timestamp(Calendar
.getInstance().getTimeInMillis()),
            "Login Session Created for "+username);
        // send auth key back to client
        return loginSessionDAO.getSessionByUsername(username)
            .getAuthorizationKey();
    } else {
        // blacklist the user again if they are detected as blacklisted
        blacklist(username,ip);
        //Notify the caller that the user has been blacklisted
        return "blacklist";
    }
} else if (activeSessionDAO.getSessionByUsername(username) != null) {
    //This block of code is used for all other actions besides those involved with
logging in and out
    //Below is an example of retrieving a number from a page, adding one, and
returning it.

    //Check that the current id and authorization key are valid
    if(checkClientID("active", username,ip,authorizationKey)){
        //Example for adding one and returning
        if(actionRequested.equals("addOne")){
            String name = (String) m.asMap().get("name");
            System.out.println("name passed: "+name);
            Integer val
            =
Integer.parseInt((String)m.asMap().get("value"));
            val++;
            return val.toString();

```

```

        }
    } else {
        //Notify the caller that the user has been blacklisted
        return "blacklist";
    }
} else {
    //This code is only executed if an invalid request is sent to the application.
    For example, a request
        //is sent to the server that has nothing to do with logging in, logging out, or
    registering, and the user
        //doesn't have an active session.
        blacklist(username, ip);
        return "blacklist";
    }

    //default return value
    return "login";
}

private String generateWhitelistKey() {
    Random r = new Random();
    String key = "";
    for(int i=0; i<10;i++)
        key += r.nextInt(10);
    return key;
}

/**
 * Performs all actions required to complete a user's session and log them out
 * @param username Username to log out
 * @return Data to be returned to the client
 */
private String processLogOutRequest(String username){

    //Retrieve the active session from the database
    ActiveSession session = activeSessionDAO.getSessionByUsername(username);

    //Check that the session exists
    if(session != null){
        //Delete the active session and log
        activeSessionDAO.deleteSession(username);
        activityLogDAO.createLog(session.getSessionId(),
            Timestamp(Calendar
                .getInstance().getTimeInMillis()),
            "Deleted active session for "+username);
    }
}

```

```

        //Add completed session to the database and log
        completedSessionDAO.createSession(session,"Session Completed on
logout");
        activityLogDAO.createLog(session.getSessionId(), new
Timestamp(Calendar
                .getInstance().getTimeInMillis()),
                "Logged out "+username);
    }
    return "login";
}

/**
 * This is the second stage in the login process. A login session has already been created,
 * and the client has an authorization key assigned. This method completes the login by
 * checking the user's credentials.
 * @param username
 * @param password
 * @param ip
 * @param authorizationKey
 * @return
 */
private String processLoginRequest(String username, String password,
        String ip, String authorizationKey) {
    // decrypt request
    //Check that the current id and authorization key are valid
    if(!checkClientID("login", username,ip,authorizationKey)){
        //notify client they have been blacklisted
        return "blacklist";
    }

    //Retrieve user from user table
    User user = userDao.getUserByUsername(username);

    //Check if password matches
    if (password.equals(user.getPassword())) {
        //Check if ip and authorization key are correct
        if (loginSessionDAO.getSessionByUsername(username).getIp()
                .equals(ip)
                && user.getAuthorizationKey().equals(authorizationKey))
        {
            //Delete login session in preparation for active session
            LoginSession session =
loginSessionDAO.deleteSession(username);
            activityLogDAO.createLog(session.getSessionId(), new
Timestamp(Calendar

```

```

        .getInstance().getTimeInMillis()),
        "Login Session deleted for "+username);

        //Check if the user is currently logged in to another computer. If they
are, delete the previous active session, log,
        //and add to completed sessions.
        if (activeSessionDAO.getSessionByUsername(username) != null){
            ActiveSession            oldSession            =
activeSessionDAO.deleteSession(username);

            activityLogDAO.createLog(oldSession.getSessionId(), new Timestamp(Calendar
                .getInstance().getTimeInMillis()),
                "Old Active Session deleted for
"+username+" because new session is being created from another machine.");

            activityLogDAO.createLog(oldSession.getSessionId(), new Timestamp(Calendar
                .getInstance().getTimeInMillis()),
                "Deleted active session for
"+username);

            completedSessionDAO.createSession(oldSession,"User logged in from another
machine");

            activityLogDAO.createLog(oldSession.getSessionId(), new Timestamp(Calendar
                .getInstance().getTimeInMillis()),
                "Logged out "+username);
        }

        //Create a new active session for the user and log
        activeSessionDAO.createSession(session);
        activityLogDAO.createLog(session.getSessionId(),            new
Timestamp(Calendar
                .getInstance().getTimeInMillis()),
                "Active Session created for "+username);
        //Reset login attempts on successful login
        userDAO.resetAttempts(username);
        //Notify user that authentication was successful
        return "WelcomePage";
    }
} else {
    //If password did not match, increase login attempt count
    userDAO.increaseAttempts(username);

    //Check if the user has passed the attempt limit. If they have, blacklist them.
    if(userDAO.getUserByUsername(username).getAttempts()            >=
Integer.parseInt(AppProperties.getProperty("maxAttempts"))){

```



```

        blacklist(username,ip);
        return "blacklist";
    }
    //Delete the login session and log
    LoginSession s = loginSessionDAO.deleteSession(username);
    activityLogDAO.createLog(s.getSessionId(), new Timestamp(Calendar
        .getInstance().getTimeInMillis()),
        "Deleting login session for "+username+" because incorrect
password");
    //User remains on login page
    return "login";
}

//Delete login session and log
LoginSession login = loginSessionDAO.deleteSession(username);
activityLogDAO.createLog(login.getSessionId(), new Timestamp(Calendar
    .getInstance().getTimeInMillis()),
    "Authentication failed: LoginSession deleted for "+username);
return "login";
}

/**
 * Blacklist the username+ip combo
 * @param username Username to blacklist
 * @param ip ip to blacklist
 */
private void blacklist(String username, String ip) {
    Session session = null;

    //Check to see if the session that needs to be removed is a login or active session
    if(loginSessionDAO.getSessionByUsername(username) != null){
        //Delete login session
        session = loginSessionDAO.deleteSession(username);
    } else if(activeSessionDAO.getSessionByUsername(username) != null){
        //Delete active session
        session = activeSessionDAO.deleteSession(username);
        //Move active session to completed session
        completedSessionDAO.createSession(session, "blacklisting user");
    }

    if(session != null){
        //Add log
        activityLogDAO.createLog(session.getSessionId(),
Timestamp(Calendar
        .getInstance().getTimeInMillis()),
        "Blacklisting user: "+username+" ip: "+ip);
new

```

```

    }
    //Notify securities
    System.out.println("finishing blacklist");
    //Add user+ip to blacklist table
    blacklistDao.addToList(username, ip);
}

/**
 * Blacklist two username/ip combos at the same time
 * @param username1 First username to blacklist
 * @param ip1 First ip to blacklist
 * @param username2 Second username to blacklist
 * @param ip2 Second ip to blacklist
 */
private void blacklist(String username1, String ip1,
    String username2, String ip2) {
    blacklist(username1,ip1);
    blacklist(username2,ip2);
}

/**
 * Checks that the username, ip, and authorization key match what the current session
 should be
 * @param sessionType
 * @param userId
 * @param authorizationKey
 * @return true if ID is valid, false if ID is invalid
 */
private boolean checkClientID(String sessionType, String username, String ip, String
authorizationKey) {
    // Encrypt Request_ID
    // Decrypt ID

    Session currentSession = null;

    //Check if current session is a login session or active session, and retrieve it
    if (sessionType.equals("login")) {
        currentSession = loginSessionDAO.getSessionByUsername(username);
    } else if (sessionType.equals("active")) {
        currentSession = activeSessionDAO.getSessionByUsername(username);
    }

    //If session doesn't exist, return false
    if(currentSession == null)
        return false;
}

```

```

//Check if ip is whitelisted
boolean ipIsWhitelisted = whiteListIpDAO.checkIfWhiteListed(ip);

//Check if id is blacklisted
boolean idIsBlacklisted = blackListIdDAO.checkIfBlackListed(username, ip);

//If the username or ip address don't match the session, blacklist and return false
if (!ip.equals(currentSession.getIp()) ||
!username.equals(currentSession.getUsername())) {

    blacklist(username,ip,currentSession.getUsername(),currentSession.getIp());
    return false;
//If the authorization key passed in the request doesn't match the session, blacklist
and return false
} else if(!authorizationKey.equals(currentSession.getAuthorizationKey())) {
    blacklist(username,ip);
    return false;
//If the id is blacklisted, blacklist and return false
} else if (idIsBlacklisted) {
    blacklist(username,ip);
    return false;
//If the ip is not whitelisted, blacklist and return false
} else if (!ipIsWhitelisted) {
    blacklist(username,ip);
    return false;
}

//ID is valid
return true;
}
}

```