



**ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY**

**SCHOOL OF GRADUATE STUDIES**

**DEPARTMENT OF COMPUTING**

Designing Service Oriented Architecture Component  
Based Development (SOCBD) Framework

**Mesay Aschalew**

**September, 2016**



**ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY**

**SCHOOL OF GRADUATE STUDIES**

**DEPARTMENT OF COMPUTING**

A Thesis Submitted to the Department of Computing of Adama Science and Technology University in Partial Fulfillment of the Requirements for Degree of Master of Science in Software Engineering

**By**

**Mesay Aschalew**

**September, 2016**

**Adama, Ethiopia**



# **ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY**

## **SCHOOL OF GRADUATE STUDIES**

### **DEPARTMENT OF COMPUTING**

#### **Designing Service Oriented Architecture Component Based Development (SOCBD) Framework**

**By Mesay Aschalew**

Names and Signature of Members of the Examining Board

**Mesfin Abebe (PhD)**

Advisor

\_\_\_\_\_

Signature

**Mr. Desalegn Abebaw**

Chairperson

\_\_\_\_\_

Signature

**Mr. Dileep Kumar G.**

Internal Examiner

\_\_\_\_\_

Signature

**Solomon Tefera (PhD)**

External Examiner

\_\_\_\_\_

Signature

## **DECLARATION**

This thesis is my original work and has not been submitted as a partial requirement for a Degree in any University

---

Mesay Aschalew

September, 2016

The thesis has been submitted for examination with my approval as University advisor.

---

Mesfin Abebe (PhD)

September, 2016

## ACKNOWLEDGEMENT

First and foremost thanks to God, being in front of me by giving hope. My gratitude goes to my advisor Dr. Mesfin without whose kindness such work would have been impossible. He gave me constructive and invaluable comments throughout this thesis work. Without the presence of his constructive, critical and friendly suggestions, this thesis would not have been Possible.

My special thanks go to all governmental and private agricultural institution network and data administrators in Addis Ababa for their support by providing valuable information regarding my research work.

I would like to express my deepest thanks to my friend, Mr. Amhare and Kelemwork who have been on my side at every time whenever I am in need of him in terms of providing me during gathering of the research data.

Finally, the acknowledgement would be incomplete without extending my deepest gratitude and respect to those people who were not mentioned here but their presence and efforts have been inspiring me. Thank you all!

## Table of Contents

|                                                        |      |
|--------------------------------------------------------|------|
| ACKNOWLEDGEMENT .....                                  | i    |
| List of Table.....                                     | v    |
| List of Figure.....                                    | vi   |
| List of Acronyms and Abbreviations .....               | vii  |
| Glossary .....                                         | viii |
| Abstract.....                                          | x    |
| CHAPTER ONE .....                                      | 1    |
| 1. INTRODUCTION .....                                  | 1    |
| 1.1. Background .....                                  | 1    |
| 1.2 Motivation.....                                    | 3    |
| 1.3 Statement of the Problem .....                     | 3    |
| 1.4 Research Questions .....                           | 5    |
| 1.5 Objective of the Study .....                       | 6    |
| 1.5.1 General Objective .....                          | 6    |
| 1.5.2 Specific Objective .....                         | 6    |
| 1.6 Methodology of the Study .....                     | 6    |
| 1.6.1 Literature Review.....                           | 7    |
| 1.6.2 Method of Data Collection and Analysis .....     | 7    |
| 1.6.3 Method of Development Environment Selection..... | 7    |
| 1.7 Significance of the Study.....                     | 8    |
| 1.8 Scope of the Study .....                           | 8    |
| 1.9 Organization of the Thesis .....                   | 9    |
| CHAPTER TWO .....                                      | 10   |
| 2.1 OVERVIEW RESEARCH CONCEPTS .....                   | 10   |
| 2.1.1 Software Engineering Methods .....               | 10   |
| 2.1.1.1 Procedure oriented Software Engineering .....  | 10   |
| 2.1.1.2 Object-Oriented Software Engineering.....      | 11   |
| 2.1.1.3 Aspect Oriented Software Engineering.....      | 11   |
| 2.1.1.4 Component Based Software Engineering .....     | 11   |
| 2.1.1.5 Service Oriented Software Engineering.....     | 15   |

|               |                                                                                   |    |
|---------------|-----------------------------------------------------------------------------------|----|
| 2.1.1.6       | Agent Based Software Engineering .....                                            | 18 |
| 2.1.2         | Framework Combination .....                                                       | 18 |
| 2.1.3         | Data Exchanging and Interoperability.....                                         | 19 |
| 2.1.4         | Agricultural Data .....                                                           | 21 |
| 2.2           | RELATED WORKS .....                                                               | 22 |
| 2.2.1         | Component Based and Service Oriented Combination .....                            | 22 |
| 2.2.2         | Interoperability and Data Exchange on Agriculture.....                            | 25 |
| 2.2.3         | Current System Challenges of the Organization .....                               | 26 |
| 2.2.4         | The Requirements of the Organizations .....                                       | 26 |
| 2.2.5         | Choosing the Best System Framework .....                                          | 27 |
| CHAPTER THREE | .....                                                                             | 29 |
| 3             | RESEARCH METHODOLOGY AND ANALYSIS .....                                           | 29 |
| 3.1           | How to Build a Suitable Framework .....                                           | 30 |
| 3.2           | Implementation of System Framework .....                                          | 30 |
| CHAPTER FOUR  | .....                                                                             | 33 |
| 4             | THE PROPOSED FRAMEWORK .....                                                      | 33 |
| 4.1           | The Proposed Framework Model .....                                                | 33 |
| 4.2           | Procedures for Implementing System Using the Framework Model .....                | 35 |
| 4.3           | System Development Rules Using the Framework Model .....                          | 36 |
| 4.4           | Overview of Framework Model .....                                                 | 36 |
| CHAPTER FIVE  | .....                                                                             | 38 |
| 5             | IMPLEMENTATION AND EVALUATION .....                                               | 38 |
| 5.1           | Background of Ethiopian Agricultural System .....                                 | 38 |
| 5.2           | Current Agricultural Data Exchange System .....                                   | 38 |
| 5.3           | THE SOCBD Framework and Data Exchange System .....                                | 39 |
| 5.3.1         | The Main Components in the Agricultural Data Exchange System for SOCBD .....      | 39 |
| 5.3.2         | The SOCBD-Based Component and Service Communication .....                         | 41 |
| 5.3.3         | The Entire SOCBD Framework Architecture for the Data Exchange System.....         | 44 |
| 5.3.4         | SOCBD Based Component Modification in the Agricultural Data Exchanging System ... | 45 |
| 5.3.5         | An SOCBD Implementation Sample Prototype.....                                     | 46 |
| 5.4           | Evaluation of SOCBD Framework.....                                                | 52 |
| CHAPTER SIX   | .....                                                                             | 59 |

|     |                                     |    |
|-----|-------------------------------------|----|
| 6   | CONCLUSION AND RECOMMENDATION ..... | 59 |
| 6.1 | Conclusion.....                     | 59 |
| 6.2 | Recommendation.....                 | 60 |
|     | Reference .....                     | 61 |
|     | Appendix 1.....                     | 68 |

## **List of Table**

|         |                                                                        |    |
|---------|------------------------------------------------------------------------|----|
| Table 1 | Participants demography.....                                           | 29 |
| Table 2 | Component service layout of web based data exchange system of MoA..... | 47 |

## List of Figure

|                                                                                                   |    |
|---------------------------------------------------------------------------------------------------|----|
| Figure 1 Component-Based Development (Adopted from [90]) .....                                    | 15 |
| Figure 2 Service-Oriented Development (Adopted from [90]).....                                    | 18 |
| Figure 3 Model of the Proposed SOCBD Framework.....                                               | 33 |
| Figure 4 Procedures for implementing the Framework model .....                                    | 35 |
| Figure 5 Communication of current data exchange system components.....                            | 39 |
| Figure 6 SOCBD-based representations of main components in the Web based data exchange system ... | 41 |
| Figure 7 Components communication architecture for data exchange system.....                      | 42 |
| Figure 8 Component repository of SOCBD framework .....                                            | 43 |
| Figure 9 Component Service registry in SOCBD framework .....                                      | 43 |
| Figure 10 The Entire SOCBD Framework for Data Exchange System.....                                | 44 |
| Figure 11 Service inserting in SOCBD Framework .....                                              | 45 |
| Figure 12 Service updating in SOCBD Framework .....                                               | 46 |
| Figure 13 Access component and its service prototype .....                                        | 50 |
| Figure 14 Component Service Registry Prototype.....                                               | 52 |
| Figure 15 Components implemented in java technology.....                                          | 55 |
| Figure 16 Components implemented in different techonology.....                                    | 56 |
| Figure 17 New Service Inseration In Access Component .....                                        | 58 |

## List of Acronyms and Abbreviations

|              |                                                           |
|--------------|-----------------------------------------------------------|
| ATA-----     | Agricultural Transformation Agency                        |
| BPEL4WS----- | Business Process Execution Language for Web Service       |
| CBD -----    | Component Based Development                               |
| CBSE-----    | Component Based Software Engineering                      |
| COM -----    | Microsoft’s Component Object Model                        |
| EIAR-----    | Ethiopian Institute of Agricultural Research              |
| EJB-----     | Enterprise Java Beans                                     |
| IEEE-----    | Institute of Electrical and Electronics Engineers         |
| MoA-----     | Minister of Agriculture                                   |
| OFIS-----    | Open Farm Information System                              |
| OSGi-----    | Open Services Gateway Initiative                          |
| SOA -----    | Service Oriented Architecture                             |
| SOCBD-----   | Service Oriented Architecture Component Based Development |
| SOAP-----    | Simple Object Access Protocol                             |
| SOFA -----   | SOFTware Appliances                                       |
| SOSE-----    | Service Oriented Software Engineering                     |
| UDDI-----    | Universal Description Discovery and Integration           |
| WSCI-----    | Web Service Choreography Interface                        |
| WSDL-----    | Web Services Description Language                         |
| XML-----     | eXtensible Markup Language                                |

## Glossary

**FILE FORMATS** that are common in computer system are JSON, XML, RDF, Spreadsheets, Comma Separated Files (CSV), Text Document, Plain Text, Scanned image, Proprietary formats and HTML.

**JSON** is a simple file format that is very easy for any programming language to read. Its simplicity means that it is generally easier for computers to process than others, such as XML.

**XML** is a widely used format for data exchange because it gives good opportunities to keep the structure in the data and the way files are built on, and allows developers to write parts of the documentation in with the data without interfering with the reading of them.

**RDF** A W3C-recommended format called **RDF** makes it possible to represent data in a form that makes it easier to combine data from multiple sources. RDF data can be stored in XML and JSON, among other serializations. RDF encourages the use of URLs as identifiers, which provides a convenient way to directly interconnect existing open data initiatives on the Web.

**SPREADSHEET** Many authorities have information left in the **spreadsheet**, for example Microsoft Excel. This data can often be used immediately with the correct descriptions of what the different columns mean.

**CSV** files can be a very useful format because it is compact and thus suitable to transfer large sets of data with the same structure. However, the format is so spartan that data are often useless without documentation since it can be almost impossible to guess the significance of the different columns. It is therefore particularly important for the comma-separated formats that documentation of the individual fields are accurate.

**TEXT DOCUMENT** classic documents in formats like Word, ODF, OOXML, or PDF may be sufficient to show certain kinds of data - for example, relatively stable mailing lists or equivalent. It may be cheap to exhibit in, as often it is the format the data is born in. The format gives no support to keep the structure consistent, which often means that it is difficult to enter data by automated means. Be sure to use templates as the basis of documents that will display data for re-use, so it is at least possible to pull information out of documents.

**PLAIN TEXT DOCUMENTS** (.txt) are very easy for computers to read. They generally exclude structural metadata from inside the document however, meaning that developers will need to create a parser that can interpret each document as it appears.

**SCANNED IMAGE** probably the least suitable form for most data, but both TIFF and JPEG-2000 can at least mark them with documentation of what is in the picture - right up to mark up an image of a document with full text content of the document. It may be relevant to their displaying data as images whose data are not born electronically - an obvious example is the old church records and other archival material - and a picture is better than nothing.

**HTML** Nowadays much data is available in HTML format on various sites. This may well be sufficient if the data is very stable and limited in scope. In some cases, it could be preferable to have data in a form easier to download and manipulate, but as it is cheap and easy to refer to a page on a website, it might be a good starting point in the display of data.

## **Abstract**

Currently IT organization experience difficulty in developing flexible software system which can satisfy constantly increasing business requirement. Although the evolution of architecture paradigms has indirect influences and impacts on software system, in the way components are designed and structured and in the way components are implemented. Component-based and service-oriented software engineering are the two solutions for implementation and deployment of large IT systems but, there are still many barriers to building a flexible system. Previously systems are developed using single architecture paradigms i.e. purely service oriented or component-based. Nowadays, , a lot of research has been done in combining the strength of different architecture paradigms in different perspective to use as one. i.e. Combining the strength of CBSE and SOSE.

In this study we are proposing service oriented architecture component based development framework design which solving current problems related to developing flexible and interoperable systems. What should be considered while combining the two software engineering approaches, and design a Service-Oriented Architecture Component-Based Development framework by performing literature review. Additionally, illustrate the designed framework by developing data exchange system for ministry of agriculture.

The purpose of this study is to design a framework which combines the features of both service-oriented and component-based software engineering to resolve automatic composition problem of SOA. To evaluate the designed framework we considered software flexibility metric namely automatic composition and loosely coupling.

Finally, the designed framework provides advantage to software developer and system easily to modify. System developed using the designed framework its components and services are loosely coupled which provide flexibility features.

**Keywords:** SOSE, CBSE, framework, data exchange and agriculture

# CHAPTER ONE

## 1. INTRODUCTION

### 1.1. Background

Recent experience has demonstrated that to tackle heterogeneity problem in IT system; software engineers used service oriented architecture instead of component based development. Despite the achievements in software engineering, development of large-scale, decentralized systems still poses major issues.

CBSE has evolved as a popular software development methodology since the introduction of Microsoft's Component Object Model (COM) in the early 90s. The purpose of CBSE is to develop systems by incorporating various independent yet well-defined software pieces in the name of components. Component-based software developers for distributed systems should decide on the architecture paradigms based on business requirements.

Service orientation is a trend in software engineering that promotes the construction of applications based on entities called services. Service-Oriented Architecture is a business-driven IT architecture approach that supports integrating your business as linked, repeatable business tasks, or services. The popularity of service orientation has increased with the emergence of Web services.

The evolution of these architecture paradigms has deep and subtle influences and impacts on software systems, in the way components are designed and structured and in the way component interactions are implemented. The evolution is driven by business integration needs both in terms of enterprise application integration and distributed systems integration.

Component-based development (CBD) according to [1] and then Service oriented architecture (SOA) according to [2] have been proposed as ways to build complex but adaptive and agile business information systems that provide effective inter- and intra-business integration.

The CBD platforms and technologies, such as CORBA Components, Sun's Enterprise Java Beans (EJB), and Microsoft's COM+/.NET, are now standards. The growing interest in Web

services has resulted in a number of industry initiatives to provide platform independent communication of software resources across the Internet [3]. Basic elements of the new Web services paradigm or service oriented development (SOD) are the standards for interoperability XML, SOAP, WSDL and UDDI [4].

On top of this basic interoperability protocol stack, new languages and specifications for defining the composition of services to form real-world business processes have emerged, such as Business Process Execution Language for Web Services (BPEL4WS) [5] and Web Service Choreography Interface (WSCI) [6]. However, the diverse nature of software systems mean that it is unlikely that systems will be developed using a purely service or component-based approach [7]. Therefore, the ability to combine the strength of CBSE and SOSE and use them in a complementary manner becomes essential. So far, a lot of research has been done in combining the strength of CBSE and SOSE in different perspective. This bring next step in the evolution of Service Oriented Architecture (SOA), raising the level of abstraction and addressing two issues of software development: complexity and reuse [17]. Since CBSE and SOSE keep on developing rapidly, exploring their combination potentials is still one interesting research topic. The Integration of component-based and service-oriented architecture offer added value in system development.

Governments must provide information services for citizens and to other stakeholders. As government institutions, they must share and use data about their common customers in order to work together. To achieve the requirements of service delivery, institutions need to exchange data about service provision, the common customers and the processes. Improving the level of automation of data exchange between organization to organization systems is widely regarded as a key enabler for agile interoperability and scalability in organization to organization collaborations [8].

In Ethiopia agriculture plays a major role in the country's economy. The Ethiopian Government has developed a growth and transformation plan on the agriculture and rural development sector for formulation of need based policies and strategies and capacitating the human resource skill so as to use the human resource in different agriculture sectors.

In order to effectively and efficiently deliver the SOCBD framework based system development proto typing the web based data exchange system framework for minister of agriculture provide two achievements i.e. providing how components and service composed as well as suggesting new data exchange system framework.

## **1.2 Motivation**

Nowadays, study show that fine integration and customization of models, frameworks and architectures made software development satisfying the requirements specification and flexible [9]-[12].

Additionally, researchers [12]-[15] presented possibilities of interoperability between component systems and the service oriented platforms. Also, researchers [16][18]-[20] they combined the strength of component based development and service oriented development depending on their domain specific but they recommend and we identified that their still need studying.

Beside combination of the two software engineering; Agriculture plays a significant role for economic development of Ethiopia. The Ethiopian Government has developed a growth and transformation plan on the agriculture and rural development sector. Using land resources in efficient and effective way, developing different agriculture ecological products within different zones for developing the regional capacity, entrenching specialization and diversification, coordinating agricultural and rural development proceedings, and strengthening the agriculture strategy and development in depth are also central to the growth and transformation plan. In order to effectively and efficiently deliver agricultural services related to data exchange researchers [21][22] studied to resolve problem agriculture institution facing. But, there is need to do more study on data exchange and we used as demonstration how to develop system using our framework.

## **1.3 Statement of the Problem**

Today, designing and implementing a large scale and complex system has been a challenging task. Two of the most well recognized software engineering paradigms coping with this challenge are: component based software engineering [24] and service-oriented software engineering [12][23]. The Component-based software engineering [25] provides support for building systems through the composition and assembly of software components. It is an

established approach in many engineering domains, such as distributed and web based systems, desktop and graphical applications and recently in embedded systems domains. In addition CBSE has proved to be successful for software reuse and maintainability in our case modifiability of components, but it does not address all of the complexities software developers are facing today, such as *varying platforms, varying protocols, various devices, the Internet*, etc. [26]. Service-oriented software engineering has emerged to address these problems software engineers facing. The growing demands for Internet computing and emerging network-based business applications and systems are the driving forces for the evolvement of service-oriented software engineering [12].

The service-oriented engineering not only introduces new value to the IT industry, but it also brings new limitations. These limitations can be summarized as follows [27]: *Interoperability of service*: as a service can communicate with any other service, every input parameter of each service must be validated completely. This results in the overload of the system and long response times. *Managing service messages*: a service depends on exchanging messages to communicate with other services. Even a single application can involve millions of these messages. *Low-level deployment issues*: many infrastructure issues (e.g. application management, security, and interface contracts) are not yet explicitly defined. SOSE simplify the integration of distributed systems build on various operating systems and technologies but still face challenges as automatic composition.

According to a survey of 2,165 companies conducted by Forrester Research, SOA adoption in public sectors remains low, whereas utilities, financial institutions and insurance companies show a high rate of acceptance and implementation [28]. Although there is an increasing trend in adoption of SOA in different sectors of the economy, when it comes to its implementation in agriculture has been relatively slow.

Agriculture is the important sector and key contributors to the national GDP of Ethiopia. The most frequently encountered e-Agriculture approach relates to improved information exchange and communication for the benefit of rural communities, farm households, and the rural service providers involved in the provision of agricultural, financial and communication services.

Various agricultural institutions and other governmental and non-governmental institutions require data exchange the right data at the right time from minister of agriculture but this couldn't be achieved because they didn't have well organized data system. High inefficiency observed in the agricultural institutions in terms of collaborative work.

Going back to main problem of our study problem we said SOSE simplify the integration of distributed systems build on various operating systems and technologies but still face challenges as automatic composition. However, the purpose of CBSE is to develop systems by incorporating various independent yet well-defined software pieces in the name of components. Component-based software developers for distributed systems should decide on the architecture paradigms based on business requirements.

Therefore, that was why combining the strength of CBSE and SOSE for improved quality attributes of software solutions is still one interesting research topic [12] and we are working on to solve automatic composition problem of service oriented architecture by providing well-defined interfacing behavior of component based development. Additionally, minister of agriculture is of the governmental office whose role to manage economy back sector of the country agriculture. So, this office or sector needs to have well organized data exchange system than currently. Developing web based data exchange system framework based on business requirement to provide data to collaborators of the sector.

## **1.4 Research Questions**

Our research study concerns in designing hybrid system development framework which solve the drawback of service oriented software engineering. To fill this gap, the features of current system development approaches are discussed firstly namely SOSE and CBSE. Thereafter, the gaps in existing system development approaches are analysed based on state-of-the-art analysis. Next, a hybrid system development framework is developed. Finally, the advantages of the proposed framework are proved by implementing web based data exchange system for MoA.

Research question which should be answered are the following: RQ#1 Does the designed framework solves the automatic composition problem of SOA? This research question answered how the framework is flexible in dynamic modification. RQ#2 Does the framework provides new advantages to system development?

## **1.5 Objective of the Study**

### **1.5.1 General Objective**

The general objective of this study is designing service oriented architecture component based development framework which fill the automatic composition challenge of service oriented architecture by integrating component based development with service oriented architecture orchestration.

### **1.5.2 Specific Objective**

The following are specific objectives of the study.

- ❖ To conduct a survey of existing approaches to service and component composition methods; and
- ❖ To design a service-oriented component based framework that integrates the component-based development and service-oriented architecture into a unified approach.
- ❖ To explore data exchange challenges in MoA and its collaborators
- ❖ To design web based data exchanging system using service oriented component based framework for MoA and its collaborators.
- ❖ To implement web based data exchanging system using service oriented component based framework for MoA and its collaborators.
- ❖ To evaluate the proposed framework software flexibility metrics.

## **1.6 Methodology of the Study**

To accomplish the main goal of this study, different methods and techniques are used. Basically, research methodology is traditionally separated into two main parts, qualitative and quantitative research.

Qualitative research is about understanding behavior and reasons, questions to be asked and gathering data. But the result is not measured in a quantifiably way, it is more concerned with why and how. It is the quality, not the quantity of samples that is important. On the other hand in quantitative research the researcher tried to measure the result in an empirical way having statistics and calculations to back up the result.

Quantitative research relies on high quantity of samples to create statistics and to show trends. This study used qualitative research method. The purpose of qualitative method was to build the theoretical perspective. It served as the guideline to shape the direction of the study [29]. Qualitative methods used in this study were systematic literature review; conceptual analysis and face-to face interviews. Hence, to meet the objectives of this study, the following techniques are adopted for this research.

### **1.6.1 Literature Review**

For proper understanding of data exchange related practices, to collect necessary information related to combining service oriented and component based software engineering and for successful completion of this study we reviewed relevant literatures such as books, journals, magazines, conference papers, manuals, and resources from electronic databases like Inspec, IEEE Xplore, ACM Digital Library, CiteSeer, Science Direct, Elsevier etc. in systematically for achieving the research objective.

### **1.6.2 Method of Data Collection and Analysis**

In this study we acquire the required information from agricultural institution through face-to-face interviewing and questionnaire method. Information is gathered from IT experts, in Ethiopian agricultural institution by using interviewing and questionnaire method. To elicit requirements from experts, both structured and unstructured interview have been used. Structured interview is used for questions that have predefined answers (both closed questions and listing of items). Whereas unstructured interview is used when a question needs explanation and even the interviewee can add his/her own questions that are expected to be important to the research [30]. Based on the findings from systematic literature review, interview and questionnaire methods, we conducted conceptual analysis to web based data exchange system.

### **1.6.3 Method of Development Environment Selection**

Application development environments (ADEs) play a critical visible role in a test software framework. With these tools, the system developer design and integrates the system that takes measurements, displays information to the end user, connects with other applications and much more. It is critical to select an ADE that not only is easy to use, but also can support multiple platforms and integrate easily with measurements and control service such as drivers. There are different considerations we discussed them in chapter 3 in detail. The selection process for a

suitable programming environment for construction data exchange system which is service oriented architecture and component based development should satisfy a range of requirements identified from both a user's and systems point of view, we discussed them in chapter 3 in detail.

## **1.7 Significance of the Study**

Component-based and service-oriented software engineering and platforms are widely used for the implementation and deployment of large-scale enterprise systems. However, by their own the two software engineering approaches have drawbacks while implementing and deploying enterprise IT system. Our study design service oriented architecture component based development framework which provide the following significance.

- Filling the gap of service oriented architecture automatic composition for web service based systems by providing well-defined structure of component based development.
- Provide flexibility, configurability and efficiency for enterprise systems developers and systems: The proposed framework will combine the features of both component-based and service-oriented software engineering. Thus, this hybrid framework will take over the main advantages of component-based and service-oriented software engineering.
- Inexperienced developers will understand the theories of CBSE and SOSE.
- Future researchers will acquire valuable experience for analysing, comparing, combining and creating system development frameworks.
- MoA got a web based data exchange system framework

## **1.8 Scope of the Study**

The aim of the study focus on designing hybrid framework of component based development and service oriented architecture i.e. service oriented architecture component based development (SOCBD) framework. The scope of this study is limited to solving the automatic composition of service oriented architecture using concept of service component interchangeability behavior. The task this study tried to accomplish are proposing system development framework and illustrating the proposed system development framework.

For the time being this study lacks an adequate system testing environment which can measure the performance of automatic composition of service of component.

## **1.9 Organization of the Thesis**

This section preview document organization and a highlight of the chapters contained in the research. Chapter 1 of the document examines Introduction Provides background information to the research and introduces the problem statement and aims of the study. State of the art on component based and service oriented development methodologies and related issues are studied in Chapter 2. Also, this chapter focuses on defining terminologies and elaborating laid upon theories and facts to avoid ambiguities in the coming chapters. As well as draws attention to critical review of related component-based and service-oriented composition frameworks reviews existing component-based and service-oriented composition approaches and investigates their advantages and limitations. Research methodology used and followed to understand the current challenge and to propose the new solution for this challenges discussed in Chapter 3 .Chapter 4 put into the proposed service component framework creates a hybrid framework which integrates the features of both component based and service-oriented system development approaches. SOCBD framework implementation and evaluation: Implements the SOCBD framework in a case study of agriculture data exchanging system discussed in Chapter 5 of the document. Finally, Chapter 6 Summary and conclusion gives concluding remarks and future work dimension of the research.

## **CHAPTER TWO**

### **2.1 OVERVIEW RESEARCH CONCEPTS**

The state of art related with component based and service oriented software engineering, combining different system development frameworks, data exchanging and issues related to our study are put forward in this chapter. The chapter also set in perspective basic concepts, theories, and terminologies used in the study.

#### **2.1.1 Software Engineering Methods**

As organizations need to improve their IT infrastructure and business processes based on the needs of their customers and the demands of the business world. The best way to keep up-to-date with competitors and maintain sustainability from an IT perspective is to embrace a disciplined, innovative and coherent system development approach. In this way organizations should be able to continue to update their ICT systems development approaches.

These software development approaches are Procedure-oriented, Object-oriented, Aspect-oriented, Component-based, Service oriented, Agent based and cloud computing approach. The two of the most dominant engineering paradigms in current software community and industry as well as the two most promising approaches that enable efficient, error-free software development are component based[24] and service oriented[23] approach. They also cope up to design and implement a large scale and complex system. The selection of any of the paradigms will in turn constrain the properties (e.g. performance, extensibility, and flexibility) of the system and the way components and their interactions are implemented in the system.

##### **2.1.1.1 Procedure oriented Software Engineering**

The Procedure oriented approach use structured and top-down development approaches and think in terms of functions and procedures. A procedure-oriented paradigm focuses on developing and ordering procedures to complete specific tasks. In order to design an information system, the system's functionality must be considered first and subsequently each functional request should be achieved as a set of procedures. In a procedure-oriented information system, the procedure-oriented data has a tenacious ability to resist reusability. Hence, the storage, enhancement and modification of data are always unpredictable and uncontrolled, and consequently system debugging and testing are much more difficult.

#### **2.1.1.2 Object-Oriented Software Engineering**

The object oriented approach come after procedure oriented paradigms by retaining all the best features of procedural method. An object-oriented paradigm focuses on data types and data-type hierarchies: the basic development principle is to model objects and their interrelationships. An object is a package of information in which each object has a built-in data type. An object-oriented approach can provide data security, reusability, maintainability, flexibility and extensibility for IT systems. In practice, with object-oriented information systems, more and more classes with duplicate or unwanted functionality are added to the class library during the system design phase. This results in designers having to use special tools and class definition to estimate whether a particular class can be reused or not. Thus, classes are always difficult to manage and codes cannot be easily reused and are difficult to trace. When system designers insert a new function into a large object-oriented system, they must execute many insertions cautiously so as to avoid significant structural changes to the system's infrastructure.

#### **2.1.1.3 Aspect Oriented Software Engineering**

The aspect-oriented approach is associated with an object-oriented approach in order to structure and encapsulate system functions. It also allows the programmer to easily add new functions to the system without having to replace the previous paradigm of the system. The aspect-oriented paradigm supports various kinds of modularised structures, including object structure, block structure, inheritance and crosscutting. The aspect-oriented development method is used to fully encapsulate crosscutting concerns and combine them with the non-crosscutting concerns, which are built using object-oriented techniques. Thus, crosscutting concerns constitute the heart of the aspect-oriented paradigm. A crosscutting concern is the core system functionality which crosscuts (affects) other concerns, non-crosscutting concerns, or the entire system. Some obvious crosscutting concerns are design features, system behaviours or properties (such as authorisation, error recovery, authentication and logging), functional coverage, exception control, and architectural constraints.

#### **2.1.1.4 Component Based Software Engineering**

Software architecture is defined as “the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution” [31]. There are several architecture paradigms according to how the concept is perceived in software architecture. The Component-Based Architecture (CBA) come

after Object-Oriented Architecture (OOA) by adding trend to build an application by separating concerns into well-defined and reusable entities which are called components. A component is defined as a well-designed and reusable part: it can be integrated with other components to make more functions available for the system [32]. Szyperski [33] offers a useful general definition - a software component is a unit of composition with contractually specified interfaces and explicit context dependencies. A component has a clearly defined interface and conforms to a prescribed behavior common to all components within an architecture [34]. Szyperski [33] Component characteristics are binary unit of composition, contractually defined interfaces, explicit dependencies, independent delivery and deployment

Component-based development approach[35] provides support for building systems through the composition and assembly of software components. It is an established approach in many engineering domains, such as distributed and web based systems, desktop and graphical applications and recently in embedded systems domains. Component development and application assembly are activities that are clearly differentiated [16].

To build exchangeable software units through clearly defined interfaces. Different manufactures offer platforms like Microsoft's Component Object Modeling (COM+), Distributed COM, Sun's Enterprise Java Beans (EJB) J2EE, Common Object Request Broker Architecture (CORBA) Component Model (CCM), and Microsoft .Net. SOFA 2 [36] and Fractal [37] as representatives of the academic system even though both of them support complete development lifecycle till runtime. Examples of component models that have been developed specifically for applications to embedded systems include Koala [38], Rubus [39], PECOS [40]. Technical Drivers for component based development approach are Flexibility, Interoperability, Reliability, Adaptability, Maintainability and Integration.

Components selection plays an important role in component based development approach. Component selection decisions are often made in an ad-hoc manner. Component selection problem is the most important task in development of component-based software from testable and reusable component because the developed component-based software depends on the selected components. So there are various methods and algorithm to search the optimal set of components from repositories. Researchers and practitioners generally use the simple component selection problem and criteria based component selection problems [41]-[43]. Simple

Component Selection Problem (SCSP) is the problem of choosing a number of components from a set of components such that their composition satisfies a set of objectives [41], [42]. Criteria-Based Component Selection Problem (CCSP) is the problem of choosing a number of components from a set of components such that their composition satisfies a set of objectives and using various criteria. [41], [42]. Find the optimal set of components according to the client-specific requirement by applying these factors like; Performance, Size, Reliability, Fault Tolerance, Time Complexity [44]. Component qualification makes sure that the selected component will: execute the desired functionality; integrate easily into the structural design of new application; demonstrate the quality attributes (e.g., reliability, performance, usability) required.

The relationships among the components are identified. The properties and behaviors of the components are identified as well [45]. Constraints are always there even after components qualify for reuse into new application, such as technical and integration constraints. Composition involves integration of components into the architecture of new application. An infrastructure is required to achieve composition. Infrastructure will: connect components; provide interoperability; execute operations. Infrastructure constitutes a collection of four architectural elements to attain component composition [46]. 1) Data Exchange Model- Procedures to access and communicate data be written for reusable components e.g. drag and drop, cut and paste and API based data exchange. 2) Automation-Kinds of SW tools, subroutines and scripts can be applied to achieve interoperability between components. 3) Structured Storage- Technology hold varied data as a single component document to index, search and access components (a single data arrangement rather than set of independent files) e.g., ActiveX and DOM. 4) Architectural Models- CORBA, RMI, Microsoft COM/DCOM. From the composition point of view, component models can be divided into two categories flat component models with simple one-level composition and hierarchical component models, where components can be composed of other components.

There are many advantages of developing the applications using the component based development approach [47]. Reusable is an important advantage of developing the applications. For example CDONTS component has been provided by Microsoft Corporation to develop e-mail systems. This component can be used in similar applications. Interoperable means

intercommunicate between components. The interoperability facilitates the programmers to integrate one application with other applications. The information systems used by ATM machines of different banks are integrated with each other. One link system is the benefit of interoperability provided by component based development approach. Up gradable: Up gradation of the web applications is easy if some new component has been introduced during the life of application. There are also other technical and economic advantages.

On the other hand component based development (CBD) approach has disadvantages. The main disadvantages of component-based development are: Customization of an already developed component according to the requirements of new application is a major issue in CBD. The developers also face a problem to adapt a component to a new platform if it were not developed for that platform. The integration of a reusable component into new component is also a major problem faced by most of the developers. Security is another major concern for the developers who reuse the components available over the Internet. Efficiency of the software applications developed using CBD is also debatable. The component to be reused may have extra functionalities that may be a requirement when it was developed. The new application that does not require extra functionalities becomes less efficient because of the loading time of those functions. Generally, the success of CBSE, it does not address all complexities that software developers are facing today especially with the emergence of internet; heterogeneity of platforms and protocols, and the difficulty of locating and selecting components against system requirements, have led to the emergence of new paradigm know as Service Oriented Software Engineering .

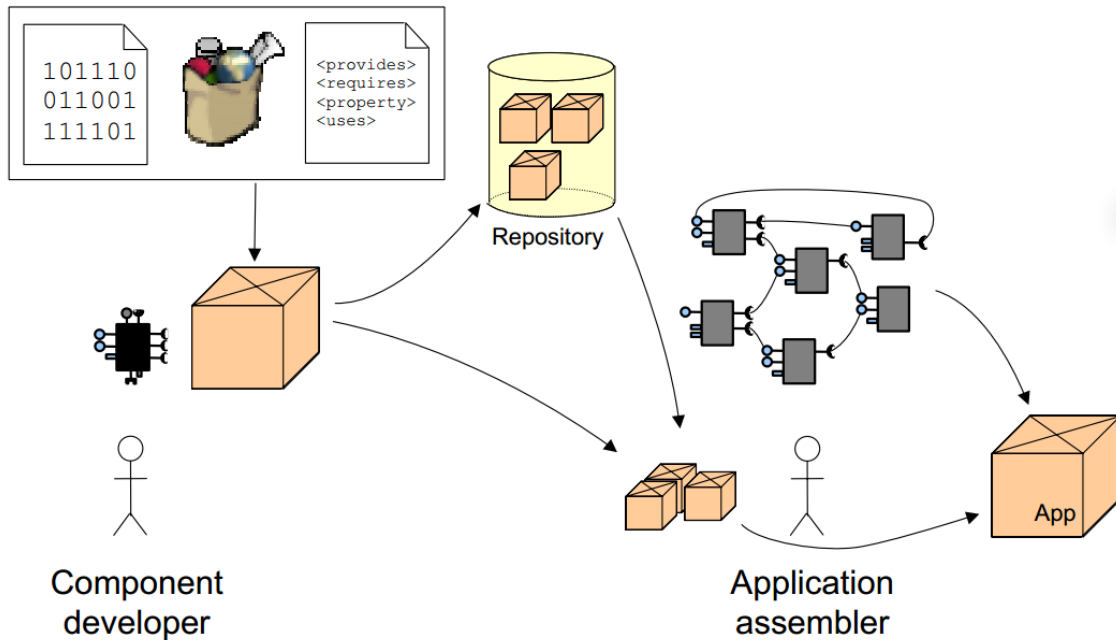


**Figure 1 Component-Based Development (Adopted from [90])**

#### 2.1.1.5 Service Oriented Software Engineering

Software architecture is defined as “the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution” [31]. There are several architecture paradigms according to how the concept is perceived in software architecture. The success of CBSE, it does not address all complexities that software developers are facing today especially with the emergence of internet; heterogeneity of platforms and protocols, and the difficulty of locating and selecting components against system requirements, have led to the emergence of new development approach know as Service Oriented Development(SOD)approach. It utilizes services as units of functionality and building blocks of an application. SOD shares the CBD idea that applications are assembled from reusable building blocks (component), but in this case the building blocks are services. Applications built by composing reusable building blocks called services. A Service is functionality that is contractually defined and contains combination of syntactic, semantic and behavioral information.

A service is generally implemented as a course-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled

(often asynchronous), message-based communication model. Service development and application assembly are activities that are clearly differentiated.

Service-oriented development helps facilitate the integration of distributed systems that are built on various platforms and technologies. Service Oriented Architecture (SOA) enables a collection of services to communicate with each other. SOA is a way of designing a software system to provide services to applications or other services through published and discoverable interfaces [48]. Service Oriented Architecture (SOA) is a set of principles that define a structural design that is loosely coupled and comprised of service providers and service consumers that interact according to a negotiated contract or interface.

SOSE technologies offer feasibility in integrating distributed systems that are built on various platforms and technologies, and further push focus on reusability and development efficiency. Web services provide much of the same functionality as CORBA and RMI, but are based on XML technology. The SOA-based systems (e.g. WebServices [49], the OSGi(Open Services Gateway Initiative) Service Platform [50]) are commonly used in industry. Web services essentially use XML to create a robust connection through WSDL, SOAP and UDDI; WSDL (Web Services Description Language): is an XML-based language for describing Web services and how to access them. SOAP (Simple Object Access Protocol): is a simple XML-based protocol to let applications exchange information over HTTP. UDDI (Universal Description, Discovery and Integration): it is a directory for storing information about web services described by WSDL. Service Bus is used to connect all participants of an SOA. Most ESB providers now build ESBs to incorporate SOA principles and increase their sales, e.g. Business Process Execution Language (BPEL).” [51].

In many ways, the terminology for services is much the same as the terminology used to describe component-based development; however, there are specific terms used to define elements within Web services: *Service*: A logical entity; the contract defined by one or more published interfaces. *Service provider*: The software entity that implements a service specification. *Service requestor*: The software entity that calls a service provider. Traditionally, this is termed a “client”; however, a service requestor can be an end-user application or another service.

*Service locator*: A specific kind of service provider that acts as a registry and allows for the lookup of service provider interfaces and service locations. **Service registry**: The service registry is an intermediary between service requesters and service providers. **Service broker**: A specific kind of service provider that can pass on service requests to one or more additional service providers.

Service composition represents the usage of a set of services to accomplish a particular task. Service composition is often considered the responsibility of service requesters. In practice, service composition is the incorporation of different services inside of an application to perform some overall function, where the application plays the role of the service requester. A service composition is written in terms of service interfaces and is considered abstract until execution time, when service providers are discovered and bound. Service compositions must handle issues relating to service discovery and service dynamics.

Service-oriented software engineering (SOSE) aims to design, develop and maintain service oriented applications in a systematic approach. Due to the complexity that is introduced when software programs as services can be discovered, composed, instantiated, executed at runtime, SOSE poses new challenges. These challenges are dynamic composition, Security challenges - loosely coupled environment, Performance - XML brings robustness not speed, Optimization, Organizing the services – registry & repository, Finding the right services and right interfaces and Transaction management is complex in interactions between logically separate system [52].

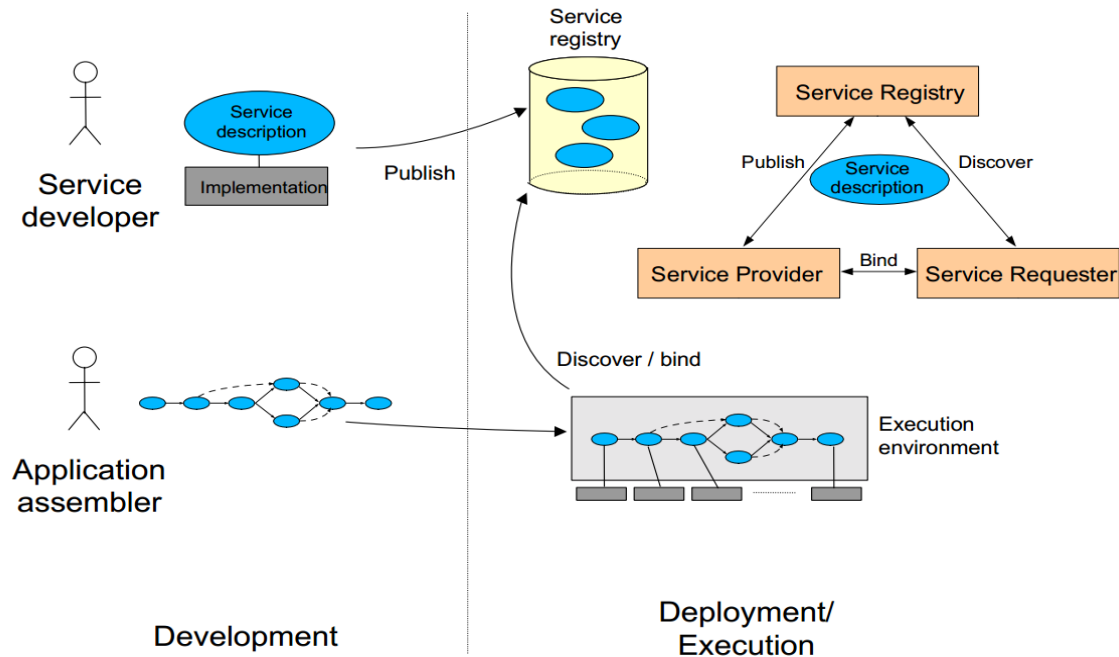


**Figure 2 Service-Oriented Development (Adopted from [90])**

#### 2.1.1.6 Agent Based Software Engineering

The agent-based approach is based on hierarchical software agent technology. In an agent-based environment, an agent may contain a group of components which are used to provide certain intelligent system functions, while various agents are classified in terms of the functionalities they offer. Thus, the agent-based system development approach can furnish self-organising, self-adapting and self-learning features for information systems.

#### 2.1.2 Framework Combination

About 80% of software engineering deals with changing existing software. Time to market is an important competitive advantage incorporate successful innovations quickly. Systems should be built to facilitate change for easy removal and addition of functionality. Recently, studies show that to tackle this change using only one software development framework is not enough so there is need to combine strength of different development framework.

The component-based and service-oriented software engineering are similar to each other in many ways [12]. To make systems, which employ them, interoperable is thus a straight-forward idea. Interoperability is a property referring to the ability of diverse systems to work together, exchange data and make use of the data that has been exchanged. This is achieved by having a common set of exchange formats, interaction patterns and using the same protocols.

Just by applying a Web service layer on top of legacy applications or components does not guarantee true SOA properties, such as business alignment, flexibility, loose coupling, and reusability. Instead, a systematic and comprehensive approach and development is critical importance, taking into account the business requirements and following recommended practices.

It would be necessary to mix the strength of both engineering to address issues that each paradigm can't solve alone. Since CBSE and SOSE keep on developing rapidly, exploring their combination potentials is still one interesting research topic.

### **2.1.3 Data Exchanging and Interoperability**

Government institutions need to collaborate in order to improve citizen service delivery. Institutions need to work across portfolio boundaries to find collaborative, networked and multi-communication channel approaches to deliver information and services from their site to citizens [53]. Development of an interoperable framework provides an integrated service by articulating a set of agreed policies, principles, and standards to facilitate information and transactions from government to all stakeholders. Improving the capability of cross institutions means communication is confidently managed, transferred and an exchange of information and data [54].

There is wide agreement in administrative practice and research that the use of ICT will only lead to savings and improvements if business processes are reorganized in order to allow for a seamless exchange of data between all agencies involved.

"Organisation interoperability, as the name implies, is the ability of organizations to effectively communicate and transfer (meaningful) data (information) even though they may be using a variety of different information systems over widely different infrastructures, possibly across different geographic regions and cultures. Organizational interoperability depends on successful technical, syntactical and semantic interoperability". According to the Institute of Electrical and Electronics Engineers (IEEE) [64], interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged.

In order to achieve interoperability, institutions must communicate effectively. For service consumers to communicate with the service provider a communication channel must be

established first and there must be a communication media on which institutions can exchange data or information using shared services. In order to communicate within the institutions, the shared service must be clearly understood and use same understanding level for information exchange [71].

Only the presence of a communication channel and data or information exchange between institutions that have shared service will not be enough; the institutions must be able to use the shared services [72].

For effective use of shared services, the institutions must have service level agreement [73]. Interoperability can be classified in to four layers on implementations; these are [74]; Organizational, Semantic, Syntactic and Technical layer Interoperability. Purpose of interoperability Enables two systems to operate together as one and uses Interaction standards like SMTP or SOAP.

Enterprise interoperability is the ability of an organization to interact with other organizations and to exchange and utilize information that has been exchanged [75]. This is achieved by having a common set of exchange formats, interaction patterns and using the same protocols. As described by [76] meaning of information sharing and standardization with the following words: “Companies must be prepared to share standards-based data free of charge. Sharing information between trading partners will result in an improved information flow and, as a consequence, improved collaboration to better serve the consumer. A resulting collaborative information platform could become the basis for further supply chain solutions... “. To enable smooth exchanges in a supply chain such as the agricultural sector, the information in the form of data must be well defined for it to be portable and understood by all stakeholders. In data exchange, data must be structured based on some harmonized schema for tasks that require data to be transferred between existing, independently created applications [77].

Gartner has estimated that over 80% of data exchanged between organizations are transferred via files such as office documents, PDF, CSV, media files, and for the majority are images and financial transaction files [78]. An SOA is made up for loosely coupled services which interoperate. An SOA supports adding new services without going back to the drawing board.

According to the business industry's perspective, the Data Interchange Standards Association (DISA), the not-for-profit group that oversees the development of EDI standards in the United States, defines EDI as [79] “the computer-to-computer exchange of business data in standard formats. XML, with its ability to transfer structured data along with meaning over the Web, immediately attracted the attention of people and organizations struggling with these issues of traditional EDI. Enterprises want to exchange data. Industry specific standards efforts are defining XML Data Type Definitions (DTDs). These DTDs represent the semantics and format of the data to be exchanged.

#### **2.1.4 Agricultural Data**

The Agricultural Economy holds 80% of the country's rural development program. The Agricultural Economy holds 45% of the country's annual income. 90% of the country's export income is from agricultural products. 85% of employment opportunity relies on agriculture. Therefore, the Ethiopian government will raise 15% of the country's annual budget. This clearly shows that the government of Ethiopia has wide attention for the agricultural sector [56].

The Ethiopian government is working on implementing e-government applications for 22 Ministries and Agencies, out of which Agriculture is one of the most focused sectors on which the government gives much attention since agriculture is the main backbone for Ethiopian Economy [57]. E-government application is not enough to achieve required goal, so additional data exchange framework is need to support the e-governance.

Unfortunately, existing information systems for agricultural data exchange lack standardization, which was obstacle for efficient exchange of information [59]. There are many types of agricultural data such as local weather, soil information, office documents, CSV and so on. Steinberger et al. [58] pointed out that the heterogeneous data structures of many agricultural systems used for data acquisition, GIS-based field indexing, different documentation tasks, and precision farming applications produce a variety of data formats and interfaces. Schmitz et al. [60] stated that a standardized system for data exchange offers new possibilities for information directed agricultural production to increase sustainability and minimize negative effects on the environment. Most standards use extensible markup language (XML) as a data format and can be implemented as web services.

Some researches on data exchange between agricultural systems have been conducted. Murakami et al. [61] proposed an agricultural system infrastructure based on service-oriented architecture. The infrastructure includes reference architecture, a standard language for data exchange between systems and services based on XML, and a service bus for use as a message-oriented middleware for connection with web services.

Kaivosoja et al. [62] developed a task controller prototype employing ISOBUS-compatible data messages. The controller utilizes external data from multiple sources during a spraying operation. Most agricultural standards for data communication have not gained a large market share, and are largely national efforts (Nash et al.) [63]. In addition, no standard embraces all agricultural domains.

## **2.2 RELATED WORKS**

### **2.2.1 Component Based and Service Oriented Combination**

Recent experience has demonstrated that to tackle heterogeneity problem software engineers use service oriented architecture instead of component based development. The challenges that are related to business benefits and chooses component based software engineering (CBSE), the ancestor of SOSE, as the benchmark. A lot of work, such as research roadmaps [65], methodologies [66], and blueprints of service engineering [67], has been proposed by researchers to investigate SOSE challenges. In particular [68] points out collaborative issues and problems caused by the dynamic nature of service-oriented system. An overview of the difference between CBSE and SOSE is given in [69].

In order to increase success of SOSE most of studies add CBSE as benchmark; because both engineering share a number of common properties. They are built around similar concepts and share many characteristics, each paradigm has its own philosophy, abstraction, issues and challenges. It would be beneficial to combine the strength of the two engineering to simplify the architecture and design of large-scale and distributed systems. Both are interface based (i.e interfaces specify the access points of components or services.), they share the same [12] objectives and principals as modularity, reusability, rapid development of software system and maintainability.

Building business applications using purely component based or service oriented software engineering is not sufficient. From the conceptual view, the functional interfaces of components and service interfaces are interchangeable and therefore there is no problem in mixing both interface types within a single component. In this manner, the component can incorporate services and provide them along with the functional interfaces for other components to use.

The implementation is provided either by the component itself or by the corresponding service providers [70]. The main difference between CBD and SOA is in the composition of components and services. The component composition is a process of assembling components together using connectors or glue code to form an assembly, a larger component or an application. In component systems, bindings between components are usually static and explicitly described whereas in service-oriented systems bindings between services are dynamic and the service composition is done during runtime when the service providers are discovered and bound (i.e. differences exist in mechanisms, approaches and implementations).

Therefore, the integration has to deal with the inherent dynamicity and provide components with the support for dynamic service availability and discovery, particularly in a sense of service and component bindings. Recent studies have coupled the strength of CBSE and SOSE, although they focus on different objectives, principals are the same, we present briefly studies related to our work.

K. Masek et al. [70] studied bridging the component-based and service-oriented worlds: in this framework, the SOFA 2 is a component system employing hierarchically composed components and other features like behavior specification using behavior protocols, automatically generated connectors supporting seamless and transparent distribution of applications, explicitly separated control part of components, and distributed runtime environment with dynamic update of components and the OSGi platform is dynamic and service-oriented bundles can be managed at runtime, without having to restart the Framework. Integration tool they choose Apache Felix as the OSGi runtime implementation. SOFA 2 components access OSGi services through dedicated interfaces, both provided and required. The integration in their approach works in both directions, i.e. allowing SOFA 2 components to use OSGi services almost transparently as other ordinary SOFA components and also, vice-versa, allowing OSGi services to use SOFA components as other OSGi services.

M. Desertot et al. [81] studied FROGi: Fractal components deployment over OSGi: which is an extension of the Fractal component model that supports deployment features. Which has been proposed for two reasons; the first one is to offer a flexible component model to the OSGi's developers to simplify bundles development. The second reason is to leverage the OSGi's deployment capabilities to package and deploy Fractal components.

FROGi is implemented on the top of the OSGi Platform by combining with Julia, the Java-based reference implementation of Fractal component model. FROGi implements these concepts by combining the Julia reference implementation of the Fractal component Model and the OSGi services platform.

H. Cervantes, and R. S. Hall [82] studied Beanome: A Component Model for the OSGi Framework. Introduce Beanome, a lightweight layer implemented on top of the OSGi framework, that implements a simple component model and framework for creating OSGi applications. Beanome is not a component model for consumer devices per service, but is a component model for the OSGi framework, which itself is used to create applications that access and provide user-level and management-level interfaces to consumer devices. The component framework, called the runtime, introduces two main concepts: component factories and a component registry. In Beanome, OSGi bundles are used to deploy COM-like components. Moreover, the OSGi service registry is used to publish components factories when the bundle is activated. A benefit of registering component factories as services is that factories can be located based on the functionalities of the components they create and not only from a unique identifier as in COM. Beanome, however, does not provide support for dynamic changes.

L. Seinturier et.al [83] studied Reconfigurable Service Component Architecture (SCA) Applications with the FraSCAti Platform. Which brings runtime management features to SCA, and discusses key principles in its design: the adoption of an extended SCA component model for the implementation of SOA applications and of the FraSCAti platform itself; the use of component-based interception techniques for dynamically weaving non-functional services such as transaction management with components. The FraSCAti platform is implemented in Java.

However, the integration and combining strength of SOA and CBD still needs studying modification of components at runtime and making components platform independent by

introducing service oriented architecture deployment. In generally, our intention was to developing a framework having characteristic of service oriented architecture and component based development. The relation between previous works is that the integrating CBD and SOA but what we are doing is designing system development framework.

### **2.2.2 Interoperability and Data Exchange on Agriculture**

Interoperability between institutions can be achieved with common understanding among different institutions working together. That means institutions must have rules, regulations, and governance in order to work together. Since working together means sharing information based on the specified rules and regulations stated in the governance, those institutions must act accordingly. Recently some studies are done in agricultural institution and its collaborators in interoperability and data exchange, although they focus on different objectives, goals are the same, we present briefly studies related to our work.

A. Tekalegn and F. Abera [85] studied Achieving Organizational Interoperability for Agricultural Research with Cross Organizational Business Process Modeling: business process modeling across multiple organizations is investigated within different governmental institutions taking part in agricultural research activities in Ethiopia. The research showed the possibility of business process modeling across multiple collaborating institutions as long as they share common goals and strategies for institutions taking part in agricultural research activities in Ethiopia.

J. Y. Kim et al [86] studied Open farm information system data-exchange platform for interaction with agricultural information systems (OFIS). OFIS data-exchange platform consists of two services: data-collection service and data-sharing service. For these services, data collection procedure for external data was analyzed and data sharing interface used to access internal data stored in OFIS was designed. In order to evaluate the data-exchange platform, two use cases were illustrated and a data consistency test was conducted. Three popular Korean OpenAPI services, which are related to agriculture, were selected for use in this study. Data groups for sharing are identified. Agricultural data involve not only environment and crop information but also financial, biological, and technical information.

R. A. Stegwee and B. D. Rukanova [87] the standards available with respect to the contribution they make toward interoperability within a specific domain. We limit our notion of domain interoperability to an organizational setting in which two organizations communicate with respect to objects and transactions within a specific or within a specific business function. This is the kind of interoperability that is the focus of many business-to-business relationships, which seem to lend themselves to substantial efficiency gains when supported by modern e-Business technologies.

As discussed in section 2.2.1 and 2.2.2 in general our intention in this paper is to show how to develop service oriented architecture component based development framework for agricultural data exchange system.

### **2.2.3 Current System Challenges of the Organization**

According to interview and questioner undertaken with the institutions to know current system challenges of the institutes the stakeholders answered the question what your institution current challenges according data exchange? According data exchange we only exchange through email and we have the data collected under our institution but there is no infrastructure to disseminate our data to others organizations. We exchange different data type with different organization. Data exchange we use email with outsiders and system developed by our developers for internal but the problem is email is sufficient and the internal system is not flexible while there is change in requirement.

### **2.2.4 The Requirements of the Organizations**

According to interview and questioner undertaken with the institutions to know current system challenges of the institutes the stakeholders answered as described in section 4.1. Also the respondents or stakeholders answered the question what would you suggest or want the data exchanging system to be? If we have infrastructure and data exchanging system we can disseminate our data to others also we will get data from them. If we need data exchange system that responds to different data users on different data types. If we have data exchange system that flexible to add and remove prerequisite for accessing the data for internal and external user.

As discussed in section 2.2.3 and 2.2.4 the current challenge of the data exchanging system and requirement of the business or institution are identified. We propose data exchange system should satisfy those requirements of MoA.

### **2.2.5 Choosing the Best System Framework**

As defined in ANSI/IEEE (American National Standards Institute/Institute of Electrical and Electronics Engineers) Standard 1471 -2000 (quoted by Minoli D. [88]), a framework is “the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution”.

According Bass L. et al [89], System framework is a high-level description of the system: Well-designed frameworks should describe the overall structure of a system and consider all of the requirements, which also include non-functional requirements, such as reusability, maintainability, scalability, and flexibility.

In recent years, information systems have become major investments for most organizations. As the blueprint of an information system, system framework represents various components which structure the entire organization and illustrates the relationship and interactivity of those components. The goal of a system framework is to build a standardized information system which assigns IT resources optimally to best support the business strategy. Furthermore, system framework can enhance cooperation among business units, share common solutions for system development across the entire organization, and provide for the reusability of existing IT components. Finally, system framework makes information systems more effective, more responsive, and cheaper [88].

The best approach to resolving the abovementioned problems is to construct an appropriate system framework. Such a framework must have the ability to estimate the current and future status of the company.

As we discussed in section 2.1.1 software development frameworks mostly used by developer for developing business systems are component based system development framework and service oriented system development framework. Also as we discussed in section 2.1.2 currently its known developing software using only one system development framework was not efficient

to handle need of today's enterprises namely service oriented and component based system development frameworks.

As discussed in section 2.2.1 studies show that combining the two development frameworks was needed to tackle challenges today's enterprises. As discussed in section 2.1.3 and 2.2.2 organization or enterprises currently need partnerships frameworks for working together to satisfy their service user or customers.

As discussed in section 2.1.4 and 2.2.2 agricultural data and partnership between agricultural data exchanging enterprises or organizations; there are different types of agricultural data and developing partnership framework for this sector is more domains specific.

As discussed in section 2.2.3 and 2.2.4 current application environment challenges and requirements of the application environment stakeholders respectively. Finally, we proposed service oriented architecture component based development framework as well as Web based data exchange system for MoA.

The service oriented architecture component based development framework use component based development for business process which having dynamic modification also uses service oriented architecture for flexibility and platform independency.

## CHAPTER THREE

### 3 RESEARCH METHODOLOGY AND ANALYSIS

In this chapter we will elaborate on our choice of research methods, identified in chapter 1. We will present a short understanding of our choice of the research methodology. In addition to this we will elaborate on the choice of data collection methods used throughout this thesis. This chapter will give a broader insight on how we have utilized the research methodology and the data collection methods to develop our thesis. We will also compare the different research methodologies available in order to get an in depth insight on the research method chosen.

Our guiding research questionnaires are open-ended which would explore the challenges and solutions in data exchanging within agricultural institutions and collaborators. Since the case for our study is explorative, the research followed by qualitative approach and used in-depth interviews to gather data from a selected group of participants. This approach would allow the author in-depth understanding of the topic. Qualitative methods used in this study were literature review; conceptual analysis and face-to face interviews.

Research setting for conducting this research, we contacted different institutes in Addis Ababa which request data from MoA. The aim was to select participants for evaluating the ability of their organizations on data exchange and exploring the challenges and solutions that they are facing during their application environment. The following table is showing the participants demography.

Table 1 Participants demography

| Name of institute          | Respondents involved | Responsibility |
|----------------------------|----------------------|----------------|
| ATA                        | 1                    | IT manager     |
| EIAR                       | 1                    | IT manager     |
| MoA                        | 1                    | ICT Director   |
| ECX                        | 1                    | IT Manger      |
| Central Statistical Agency | 1                    | IT Manger      |
| MoFED                      | 1                    | IT manger      |

### **3.1 How to Build a Suitable Framework**

A working definition of a partnership is “a collaborative relationship between entities to work toward shared objectives through a mutually agreed division of labor.” To develop framework which satisfy the requirement of the organization? Firstly, it is necessary to defining the need for a partnership or institutions. Secondly, we will proceed designing the flow structure of components and service. Finally, we will implement the flow logic designed.

To build a suitable framework we define the suggestions and need of institution provided by their IT experts. According to interview and questioner undertook with the institutions to know the need of their institutes or what they require the system to be; the stakeholders answered as we discussed in section 4.2. Additionally, which research concept should be taken into consideration to provide solution?

### **3.2 Implementation of System Framework**

The selection process for a suitable programming environment for construction agricultural data exchange systems should satisfy a range of requirements identified from both a user’s and systems point of view. Software developer must consider different selection criteria while selecting system development environment. We considered the power, real time operation, large software development, integration, portability, security and familiarization. Additionally, we proposed web service based data exchange system.

We selected web service based data exchange system as solution based different reason. A web service has special behavioral characteristics: XML-based: By using XML as the data representation layer for all web services protocols and technologies that are created, these technologies can be interoperable at their core level. As a data transport, XML eliminates any networking, operating system, or platform binding that a protocol has.

Loosely coupled A consumer of a web service is not tied to that web service directly; the web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must also be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems. Coarse-grained

component based technologies such as Java exposes their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level. Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service. Businesses and the interfaces that they expose should be coarse-grained.

Web services technology provides a natural way of defining coarse grained services that access the right amount of business logic. It supports Remote Procedure Calls (RPCs): Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.

Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments over the past couple of years. Both technologies are distributed and accessible through a variety of RPC mechanisms. A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component. Supports document exchange: One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be simple, such as when representing a current address, or they can be complex, representing an entire book or RFQ. Web services support the transparent exchange of documents to facilitate business integration.

According to these selection criteria we used java programming language and XML to implementing the SOCBD framework for the agricultural data exchange system as well as BPEL (Business Process Execution Language) is used to establish the link between partners. XML used for providing platform independency while services of partners communicate and also organizations want to exchange data.

Industry specific standards efforts are defining XML Data Type Definitions (DTDs). These DTDs represent the semantics and format of the data to be exchanged and BPEL used for orchestration which deals with the interactions between services.

The other thing we will perform to evaluate the SOCBD framework based data exchange system. When evaluating one framework researchers most of time they consider different evaluation criteria into account which shows their studying area scope. However, according to our studying scope we considered firstly how the framework is flexible for insertion and updating new component or service, secondly what advantage the framework brought for organizations or enterprises when they need data exchange system for service delivery to requester.

## CHAPTER FOUR

### 4 THE PROPOSED FRAMEWORK

In section 2.2, the features of current component-based and service-oriented combination methods were discussed. In this chapter, a proposed integrated framework is presented in the form of a data exchange system for agricultural mister and its collaborators.

#### 4.1 The Proposed Framework Model

The model attempts to clarify the entire structure and development processes required to produce the proposed framework. The aim of the conceptual model is to depict a scene in which all the basic building blocks that form part of the future framework are extracted.

The model consists of all of the building blocks that should to be taken into consideration when designing a framework. In order to design an optimal framework, one has to consider all of the basic elements of the component based development and service oriented architecture environments of the organization under consideration and an integration bus which connects/combines the two worlds.

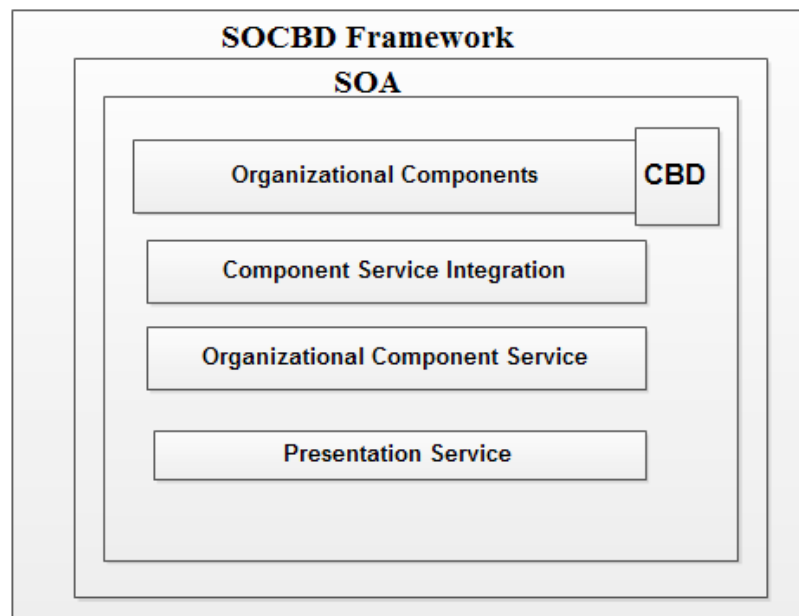


Figure 3 Model of the Proposed SOCBD Framework

As show in Figure 4.1 the framework model consists of all of the building blocks that should be taken into consideration when designing the SOCBD framework based system. In order to develop system using the framework one has to consider the basic concepts and elements of the component based development and service oriented architecture. Additionally, those considerations should be taken based on working environment of the organization or business.

The SOCBD framework model consists of two main parts namely component based development environment and service oriented architecture deployment.

The component based development environment of the framework develop the business or organizations process and working rule followed under the organization or business between organization or business which having dynamic modification.

The service oriented architecture of the framework deployment the components of the organization which developed using component based development in perform of flexible and platform independent architecture. Components provide services to other components, and consume services, too. Components can be anything that can provide services or use the services offered by other components.

According to the conceptual model, the proposed framework should be designed with the following development principles as a basis:

- Dynamic system update and modification: The main elements should be modified, deleted and updated at system runtime.
- Manageability of component and service: The components and services must be manageable in order to reduce the complexity of the system.
- Flexible component and service integration method: The component and service can be combined with various system functionalities and requirements as a basis.
- Loose coupling between integrated components: To satisfy the change of requirements, the proposed framework should support the loose-coupling mechanism.

- Independent of component/service vendor and programming language: In order to enhance the flexibility and compatibility, the components and services can be offered by different providers and be developed in various programming languages.

## 4.2 Procedures for Implementing System Using the Framework Model

As discussed in section 5.1 the task of the conceptual model is to extract services and components from existing systems and integrate them in a uniform manner. Thus, organizational requirements should first be analysed in order to match the application environment. The next step is that the required components and services must be identified and specified with their relationships established: this would represent the SOCBD framework that could finally be implemented to achieve the required system functionalities. It follows that five major steps would be required for the implementation of a framework. These are summarized as follows:

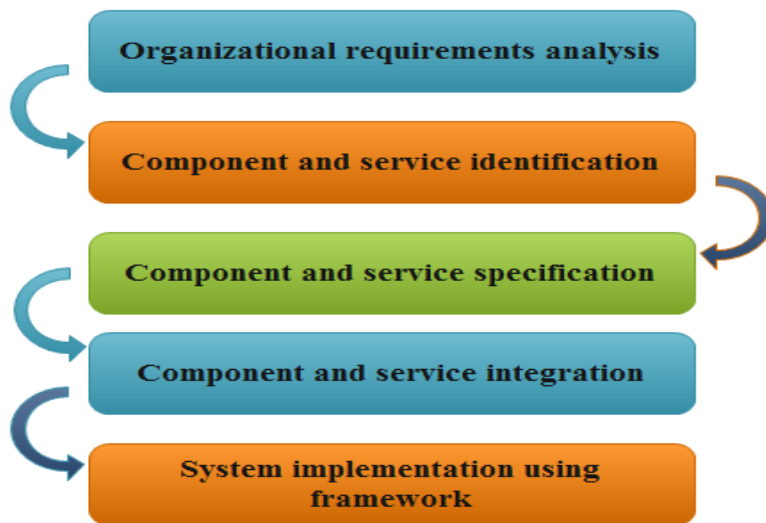


**Figure 4 Procedures for implementing the Framework model**

As show in figure 4.2 firstly current system application environment and organizational requirements should be analysed. Secondly, the dependencies and relationships between system components and services of organization should be identified. Thirdly, describing in detail attributes such as communication method and management mechanism of selected components and services. Fourthly, integration of the features of both the component-based and the service-oriented system development approaches. Finally, implementing agricultural data exchange

system for Ethiopia MoA using the framework. Finally, testing the system developed using SOCBD framework.

### **4.3 System Development Rules Using the Framework Model**

The purpose of SOCBD framework is to combine the features of both the service oriented architecture and component based development and also to provide flexibility in deployment and modification as need while working environment of the organization change. In order to achieve the purpose system development using the framework model should be based on the following rules:

- Organizational or business component selection based on organizations objective.
- Organizational or business process and rules should follow the component based development.
- The component developed should be published as service of the organization.
- Component repository should follow organizational physical structure.
- Accessing the component repository should be based on proxy.

### **4.4 Overview of Framework Model**

The proposed SOCBD framework uses component based development concept in order to achieve manageability and reusability and service oriented architecture used to provide platform independency functionality of systems. Components will communicate with each other component through service which provided by these components. These components to communicate as service they will be configured with relevant configuration files. These configuration file is used to describe the properties of a component and provide the required communication methods between these components. The notion and concept of “interface” is key to successful design in both component-based and service-oriented systems. As discussed in chapter 2 under section 2.2.1 studies and researchers presented some companies presently use and developed combined system development approach of the two world and platform having functionality of both worlds respectively. But, our framework provides more flexibility in development system using it. For example the type of data they exchange may change through time or security component they use may be updated any time our framework we provide mechanism to replace components property easily. The data servers of organization placed according the workflow of the organization.

For independence from programming languages the system development using our framework the java language mapping takes advantage for supporting component based notions such as property injection. For independence from interface definition language WSDL and XML are takes advantage. For independence from communication protocols the SOAP takes the advantage because our framework bases web service. For integrating the cooperating organization or business the BPEL is takes the advance for web service interoperability.

## CHAPTER FIVE

### 5 IMPLEMENTATION AND EVALUATION

In this chapter, we illustrate how to use our framework for system development using as case study developing web based data exchange system for MoA and evaluate the system developed using our framework by considering what characteristics the proposed framework added.

#### 5.1 Background of Ethiopian Agricultural System

The system is supposed to provide service to employees over the Intranet and to customers over the Internet. Thus, the Ethiopian agriculture minister and institutions currently provide the following functionalities:

- ❖ Data service for internal employee and its collaborates
- ❖ Research development and collaboration in development of research with other institutions.

#### 5.2 Current Agricultural Data Exchange System

Components can be anything that can provide services or use the services offered by other components. The working data exchange system architecture of the organization contains the data user's component, service interface component, authentication component and legacy system repository component as shown in figure 5.2. The data user's components represent data client or requesters of the organization. The service interface components represent how the data users provided and request data from the organization. The authentication component request authentication for accessing the data of the organization i.e. for portal and intranet data users requires the privilege of the system and for email data users requires privilege of the responsible stack holders. The legacy system repository component represent where the organizational data stored and provided from.

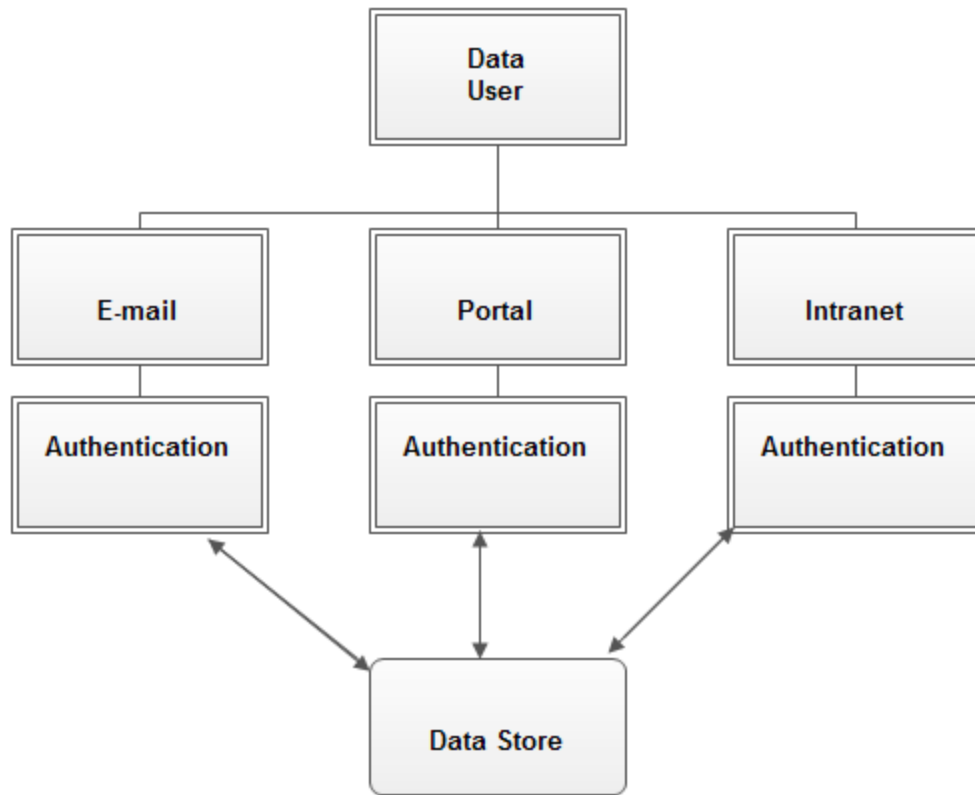


**Figure 5 Communication of current data exchange system components**

### **5.3 THE SOCBD Framework and Data Exchange System**

#### **5.3.1 The Main Components in the Agricultural Data Exchange System for SOCBD**

The proposed data exchange system consists of six main components to deliver service of the system. These components namely are data entry, business logic, access, business function, authentication and common communication component. In this section, each component is discussed in detail.

#### **Authentication Component**

This component is responsible for confirming the identity of the user and evaluating the data access permission of the user. A user account established for service users and these accounts are associated with a set of authorization identifier or attributes. Each authorized identifier includes an authorized operation and authorized file type.

Based on the authorization identifier the business files are forwarded to user. User policies determine which users can login to Data Exchange and access information in Data Exchange processes. It is Credential component for Data Exchange user accounts. User authentication controls which users can login to which page and Operation to perform.

### **Business logic component**

This component is responsible for updating and managing service user information and institution service level agreements. Business or organizations may need a change or have a change on their service level agreement. Based on their service level agreement the service user information is managed using this component. Web page used for customizing and monitors data processing partners profile. It also administers users and resources in Data Exchange. User policies determine which users can login to Data Exchange and access information in Data Exchange processes.

### **Access Component**

The access component manages processing in Data Exchange. It sends the intended page to data user for processing. After the data user performed any processes, the access component takes result and puts them in the correct location to intend for a partner. This component responsible to give any GUI change to or management service for any GUI in the entire data exchanging system based on business logic for service user to provide service for user.

### **Data Repository Component**

This component used for handling data by using the database connection or for the purpose of storing and fetching data. The Repository for Data Exchange resides in an Oracle or SQL Server database. The repository database tables contain the metadata required. When you create or modify partners and profiles. Access component reads the metadata in the repository to determine the requirements for processing documents.

### **Common Communication Component**

This component responsible for establishing Communication Bridge to the entire data exchanging system and to manage different components based on the component container. Used as an communication interface for the interaction between the above components.

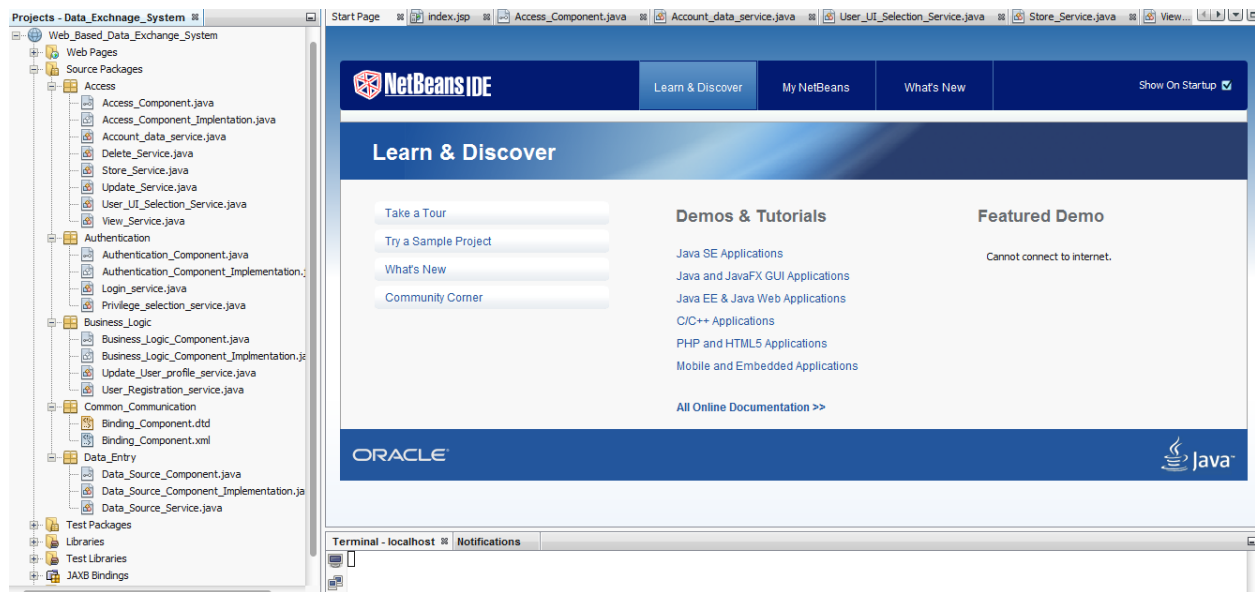


**Figure 6 SOCBD-based representations of main components in the Web based data exchange system**

### 5.3.2 The SOCBD-Based Component and Service Communication

The communication between SOCBD-based components depends on the service they provide as stated in configurable configuration files. The sequence of service requests is managed by the component service registry, which also provides a loose coupling mechanism. In the SOCBD framework, the modification of a components service relies on the relevant configuration file which can be configured to produce SOCBD-based components.

The communication between SOCBD-based components is achieved by various configuration files. Each configuration file must define at least one service of the component. Configuration files enable SOCBD-based components to communicate externally.

Each configuration file should define multiple communication protocols; this allows one SOCBD-based component to communicate with other SOCBD-based components, applications or systems depending on the different requirements and environments.

One SOCBD-based component can invoke multiple services which belong to other SOCBD-based components, applications or systems, while each component service relies on the configuration file to provide the references and properties of the component service.

Thus, in the SOCBD framework, a service can be seen as the configurable part of SOCBD-based component, and thus, an SOCBD-based component can play the role of both service provider and service consumer. The following diagram illustrates how communication between SOCBD-based components.

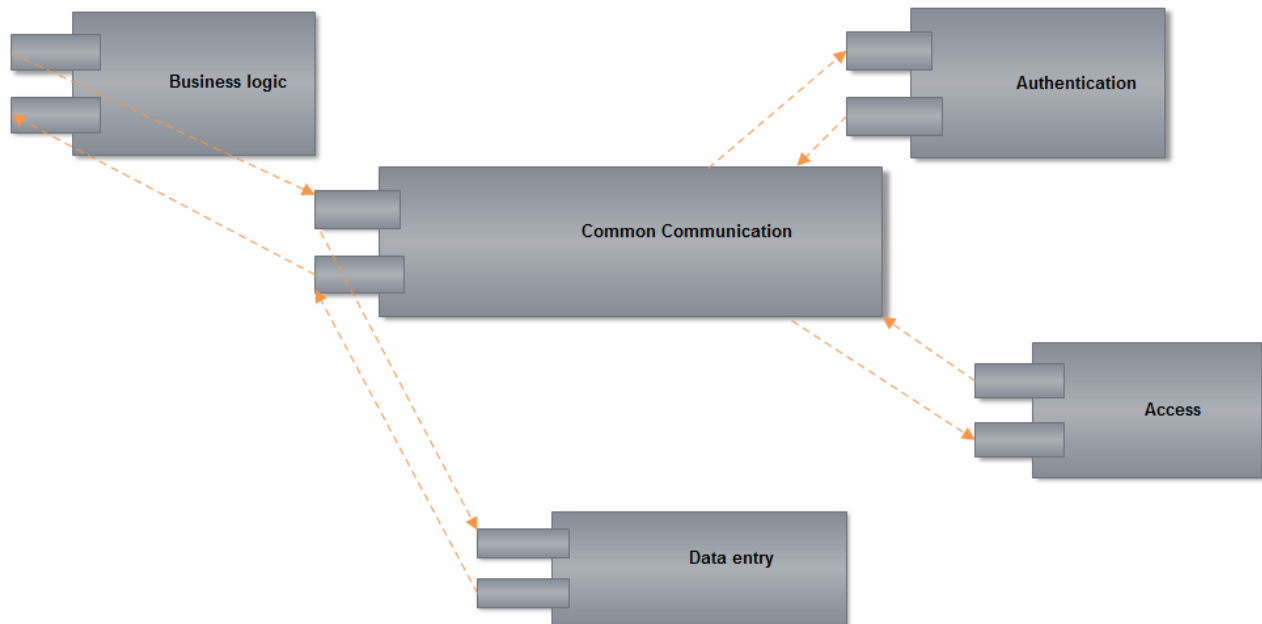


**Figure 7 Components communication architecture for data exchange system**

### Component Repository

The SOCBD-based component repository manages various SOCBD-based components and relevant services, from where they can be easily invoked and added to the system. As we discussed in Chapter 2 section 2.1 and 2.2, the existing component and service composition frameworks cannot provide such a manageable mechanism for their main building block. Thus, the SOCBD-based component repository can equip a highly manageable application environment as well as reduce the complexity of adding and modifying system functionalities.

The communication mechanism of an SOCBD framework is independent of the programming languages used to implement the various SCSD-based components. The SOCBD-based component repository can handle various SOCBD-based components defined by a number of technologies. In order to satisfy the requirements of different application environments, SOCBD-based components in the repository can also be invoked by various technologies.

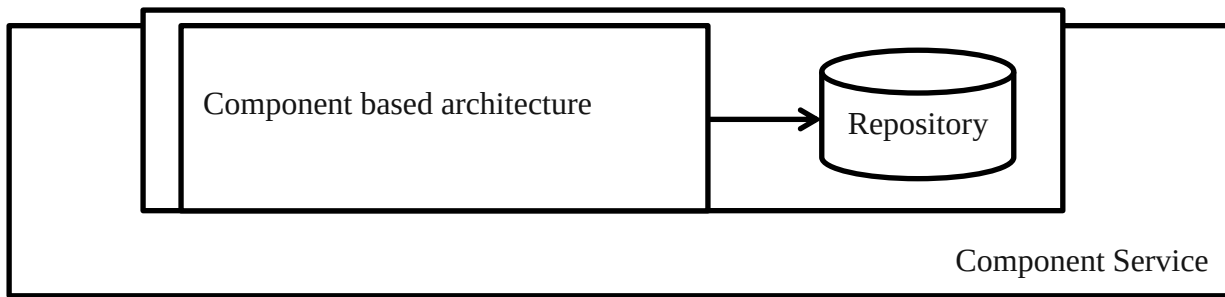


**Figure 8 Component repository of SOCBD framework**

### **Component Service Registry**

In the SOCBD framework, component implements relevant configuration files to facilitate communication with other SOCBD-based components, applications or systems. Based on this feature, many interactions will exist in the SOCBD-based system. Thus, the coordination of communications becomes an essential issue. As a result, a component service registry is required in the proposed framework.

This component service registry can be viewed as a common communication component which is responsible for managing interactions between SOCBD-based components in the framework. An interaction is a sequence of service requests which are invoked by the SOCBD-based components through the relevant configuration files. The interactions are recorded by the component service registry which is also responsible for managing the logic of functionalities and operations. Figure 5.5 illustrates the SOCBD-based component interactions by the component service registry.

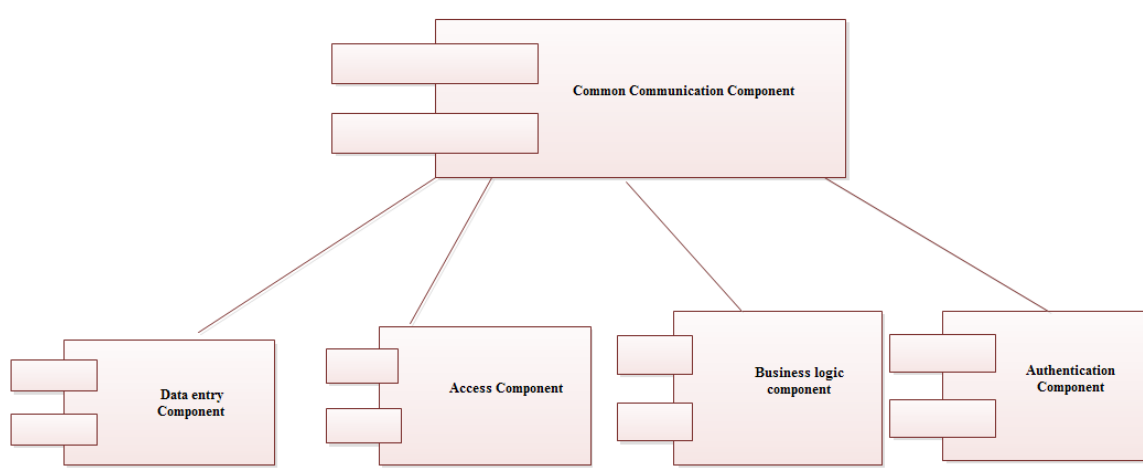


**Figure 9 Component Service registry in SOCBD framework**

### 5.3.3 The Entire SOCBD Framework Architecture for the Data Exchange System

The whole SOCBD system architecture will be shown in order to investigate the feasibility of using an SOCBD concept in the real system application environment. The given architecture uses the SOCBD layer instead of the original component layer and or service layer in order to maximize the compatibility and flexibility while system development and deployment. The SOCBD framework based agricultural data exchange system uses service oriented architecture to comprehend system functionalities, and embraces SOCBD framework components to implement the business services. The SOCBD component consumes the services offered by other SOCBD components through well-defined configuration files.

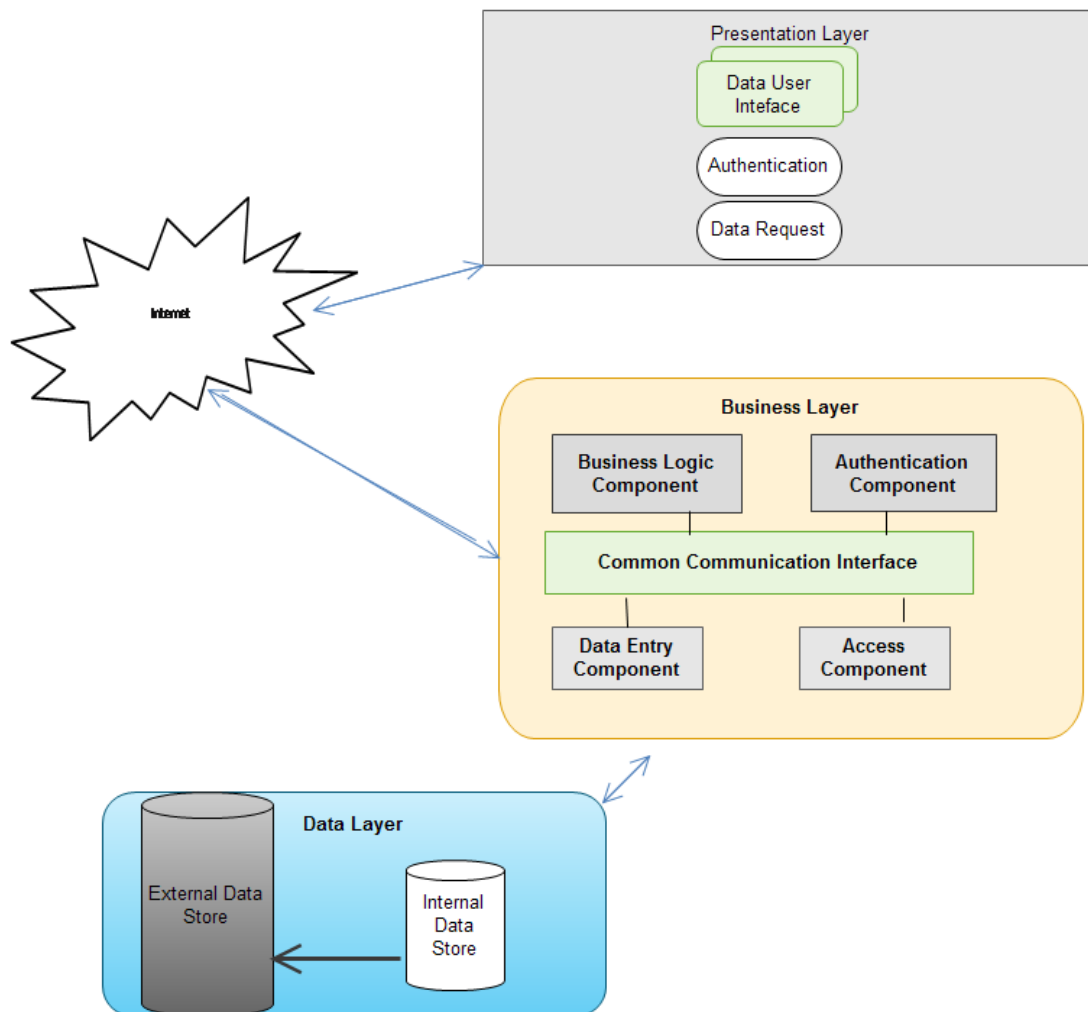


**Figure 10 The Entire SOCBD Framework for Data Exchange System**

### 5.3.4 SOCBD Based Component Modification in the Agricultural Data Exchanging System

#### SOCBD-Based Component Service Insertion

The addition of new SOCBD-based components service at runtime in the system developed using SOCBD framework. The first step is to deploy all Java objects, images and Web pages into their respective folders. Secondly, the DataBean object will build all necessary tables and databases, and the SOCBD-based component service script must be invoked for the configuration of the database requirements. In this step, the new SOCBD-based component service can be tested without intervention by the user. Finally, the module registry of the system will register the new SOCBD-based component service as a prescribed extension. If any user wants to check all of the available functions, this new SCSD-based component service will be shown in the system extension list and can be added to the profile of that user.

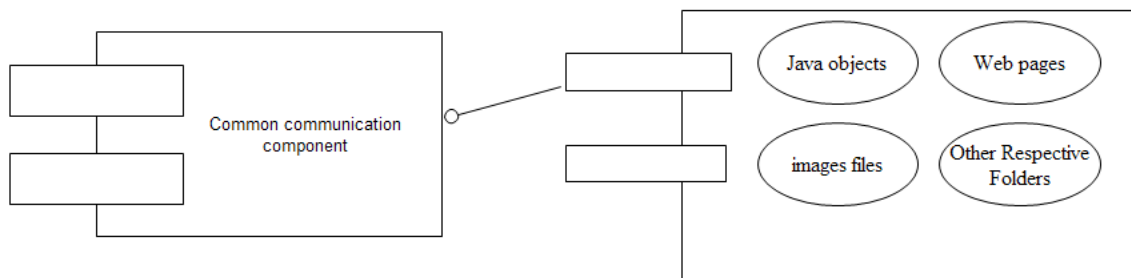


**Figure 11 Service inserting in SOCBD Framework**

#### SOCBD-Based Component Service Update

The SOCBD-based component update includes one of the following three operations:

- Add new SOCBD-based component or component service;
- Delete SOCBD-based component or component service

When updating a SOCBD-based component service in system of SOCBD framework, the following steps must be executed:

- Allow users to complete the current session with the SOCBD-based component which needs replacement.

- Disconnect the old version of the SOCBD-based component, but do not restrict the access to the new version.
- When the new version of the SOCBD-based component is installed, data integrity must be checked to ensure that all records, fields, and tables are not affected.

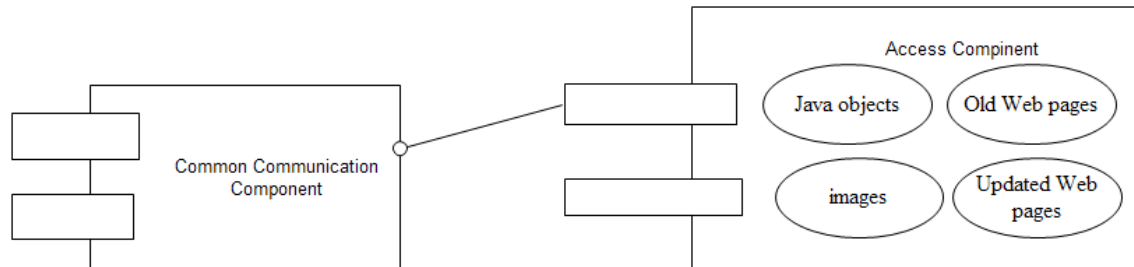


**Figure 12 Service updating in SOCBD Framework**

#### **SOCBD-Based Component Service Insert and Update Constraint**

The SOCBD framework development aims to provide flexibility, however to get this flexibility advantage there is consideration should be noticed. Those considerations are while inserting new component or service this insertion should follow common files naming the same as while updating component or service.

#### **Advantages of the SOCBD Based Data Exchange System of MoA**

Flexibility of modification, when organizations or software engineers need little or huge modification they modify their data exchange system component and service by going to the component or service particular without going to modify the entire system.

Platform independency, when organizations or software engineers need to deploy the data exchange system which developed based on the SOCBD framework. They can deploy their system or access the system in any platform which fulfills the basic requirement.

#### **5.3.5 An SOCBD Implementation Sample Prototype**

The best way to implement system using the SOCBD framework is clearly to identify the business component and service provided by that component. As well as clear identify which service the relation between service for data input and output. In the web based data exchange

system of MoA the components are previously identified now we will provide their service and their service relation.

**Table 2 Component service layout of web based data exchange system of MoA**

| Component                      | Service                     |
|--------------------------------|-----------------------------|
| Authentication Component       | Login service               |
|                                | Privilege selection service |
|                                |                             |
| Business Logic Component       | User Registration service   |
|                                | Update User profile service |
|                                |                             |
| Access Component               | Account data service        |
|                                | User UI Selection service   |
|                                | Store service               |
|                                | View Service                |
|                                | Update service              |
|                                |                             |
| Data Repository Component      | Data Source service         |
|                                |                             |
| Common Communication Component | Binding service             |

The purpose of this section is to illustrate the prototype codes of the main components of web based agricultural data exchange system in SOCBD-based framework and provide basic idea of the SOCBD framework for system developers or software engineers. As demonstrated in Table 2, the SOCBD-based architecture components of agricultural data exchange system and their service are specified clearly.

The SOCBD-based components other than common communication component they call binding service to interact with other components. The binding service provides information which component requires and provides service as well as a service description which specifies what operations can be called on the composite, including their parameters and return types. For each service the method of access can be defined.

Additionally, a composite is a container that is used for defining services, components and their references. It is used to assemble SOCBD framework based data exchange system elements into logical groupings of components. Composite is treated as a component, which is in our case common communication component. This implemented in XML structure file.

For the client JSP offers browser-based functionality to send request for service from the system and then the system calls those service and replay required service. This implemented in Web programming languages.

The SOCBD-based components service are service provider of the web based agricultural data exchange system. Services are implemented in any high level programming languages.

As describe in table 5.3.5 component and service relation of web based data exchange system the access component provide Account data service, User UI Selection service, Store service, View Service and Update service. The following code prototype show how the development and coding structure of SOCBD framework look likes.

### **Access component and its service**

```
package Access;

import Authentication.Authentication_Component.*;
import Business_Logic.Business_Logic_Component.*;
import Data_Entry.Data_Source_Component.*;

public interface Access_Component {

    Account_data_service account_validation(String username, String password);

    User_UI_Selection_Service ui_selection(int flag);

    Store_Service Store();

    View_Service view();

    Update_Service Update();

}
```

```
package Access;
```

```
public abstract class Access_Component_Implementation implements Access_Component{  
    public Account_data_service account_validation(String username, String password) {  
        Account_data_service account=new Account_data_service();  
        // the logic  
        return account;  
    }  
}
```

```
public User_UI_Selection_Service ui_selection(int flag) {  
    User_UI_Selection_Service ui_selection=new User_UI_Selection_Service();  
    //the logic  
    return ui_selection;  
}
```

```
public Store_Service Store() {  
    Store_Service store=new Store_Service();  
    // the logic  
    return store;  
}
```

```
public View_Service view() {  
    View_Service view=new View_Service();  
    // the logic  
    return view;  
}
```

```
public Update_Service Update() {
```

```

Update_Service update=new Update_Service();

// the logic

return update;

}

}

package Access;

public class Account_data_service {

    //here the logic the web and the data entry

}

```

**Figure 13 Access component and its service prototype**

## **The communication of all components**

```

<?xml version="1.0" encoding="UTF-8"?>

<composite targetNamespace="http://org.service.naming" name="bindingService">

    <component name="Access">

        <implementation.java class="Access_Component_Implentation">

            <service name="Account_data_service">

                <binding.ws uri=""/>

            </service>

            <service name="Store_Service">

                <binding.ws uri=""/>

            </service>

            <service name="Update_Service">

                <binding.ws uri=""/>

            </service>

            <service name="User_UI_Selection_Service">

```

```

        <binding.ws uri=""/>
    </service>

    <service name="View_Service">
        <binding.ws uri=""/>
    </service>
</implementation.java>

<reference name="Data_Source_Service">
    <interface.java interface="Data_Source_Component" />
</reference>
</component>

    <component name="Authentication">
        <implementation.java class="Authentication_Component_Implementation">
            <service name="Login_service">
                <binding.ws uri=""/>
            </service>

            <service name="Privilege_selection_service">
                <binding.ws uri=""/>
            </service>
        </implementation.java>
        <reference name="Access">
            <interface.java interface="Access_Component" /> </reference>
        </component>

        <component name="Business_Logic">
            <implementation.java class="Business_Logic_Component_Implmentation">
                <service name="Update_User_profile_service">

```

```

        <binding.ws uri=""/>
    </service>

    <service name="User_Registration_service">
        <binding.ws uri=""/>
    </service>

</implementation.java>

<reference name="Access">
    <interface.java interface="Access_Component" />
</reference>

<reference name="Authentication">
    <interface.java interface="Authentication_Component" />
</reference>

</component>

<component name="Data_Entry">
    <implementation.java class="Data_Source_Component_Implementation">
        <service name="Data_Source_Service">
            <binding.ws uri=""/>
        </service>
    </implementation.java>
</component>

</composite>

```

**Figure 14 Component Service Registry Prototype**

## 5.4 Evaluation of SOCBD Framework

Software flexibility is one of the properties that indicate if the software is easy to change. Flexible software can easily adapt to user requirement and/or environment changes during the software development period or after the software is deployed.

As mentioned in [91], Flexible changes have the characteristics as follows: *Change is controllable*, i.e. change is limited to some range; *Change is repeatable*, i.e. change can be carried out repeatedly at almost the same ease; *Change is inversed*, i.e. change can be restored at almost the same ease; *Change is harmonious*, i.e. change is consistent and the change at one point does not lead to failure at other points, and that is easy to change without compromising dependability; *Change is easy*, i.e. change is accomplished with little time and low cost. Metric for measurement of software flexibility are Time, cost and software flexibility.

However, to evaluate the system developed with SOCBD framework we considered software flexibility metric. The aim of the study is to make system development in component based and service oriented more flexible. To see how these feature applicable, let us see the binding service of common communication component of the agricultural data exchange system.

```
<?xml version="1.0" encoding="UTF-8"?>

<composite targetNamespace="http://org.service.naming" name="bindingService">

  <component name="Access">

    <implementation.java class="Access_Component_Implementation">

      <service name="Account_data_service">

        <binding.ws uri=""/>

      </service>

      <service name="Store_Service">

        <binding.ws uri=""/>

      </service>

      <service name="Update_Service">

        <binding.ws uri=""/>

      </service>

      <service name="User_UI_Selection_Service">

        <binding.ws uri=""/>

      </service>

    </implementation.java>

  </component>

</composite>
```

```

    </service>

    <service name="View_Service">

        <binding.ws uri=""/>

    </service>

</implementation.java>

<reference name="Data_Source_Service">

    <interface.java interface="Data_Source_Component" />

</reference>

</component>

    <component name="Authentication">

        <implementation.java class="Authentication_Component_Implementation">

            <service name="Login_service">

                <binding.ws uri=""/>

            </service>

            <service name="Privilege_selection_service">

                <binding.ws uri=""/>

            </service>

        </implementation.java>

        <reference name="Access">

            <interface.java interface="Access_Component" />

        </reference>

    </component>

    <component name="Business_Logic">

        <implementation.java class="Business_Logic_Component_Implementation">

```

```

        <service name="Update_User_profile_service">
            <binding.ws uri=""/>
        </service>

        <service name="User_Registration_service">
            <binding.ws uri=""/>
        </service>

    </implementation.java>

    <reference name="Access">
        <interface.java interface="Access_Component" />
    </reference>

    <reference name="Authentication">
        <interface.java interface="Authentication_Component" />
    </reference>

</component>

<component name="Data_Entry">
    <implementation.java class="Data_Source_Component_Implementation">
        <service name="Data_Source_Service">
            <binding.ws uri=""/>
        </service>
    </implementation.java>
</component>

</composite>

```

**Figure 15 Components implemented in java technology**

The implementation of a component can be configured based on the chosen implementation technology, e.g. Java, BPEL, spring , ccp, and .Net etc. implementation technology independency feature.

```

<Component name="Component_name">

    <implementation.spring location="SpringApplication.jar">
</implementation.spring>

//service from springapplication

</Component>

<Component name="Component_name">

    <implementation.bpel process="process_name">

        //service from bepl process

    </implementation.bpel>

</Component>

```

**Figure 16 Components implemented in different techonology**

Support loose coupling communication mechanism as demonstrated in above in the code, class Access\_Component\_Implentation contains a pointer only to Access\_Component, which is implemented as an interface in Java. Access\_Component\_Implentation classes implement this interface. If a new user requirement arises later to execute another access component service , a new class can easily be added without requiring any modification of the Access\_Component\_Implentation class. For example delete service is required in access component.

```

package Access;

import Authentication.Authentication_Component.*;

import Business_Logic.Business_Logic_Component.*;

import Data_Entry.Data_Source_Component.*;

public interface Access_Component {

```

```

Account_data_service account_validation(String username, String password);

User_UI_Selection_Service ui_selection(int flag);

Store_Service Store();

View_Service view();

Update_Service Update();

Delete_Service delete();

}

package Access;

public abstract class Access_Component_Implementation implements Access_Component{

    public Account_data_service account_validation(String username, String password) {

        Account_data_service account=new Account_data_service();

        // the logic

        return account;

    }

    public User_UI_Selection_Service ui_selection(int flag) {

        User_UI_Selection_Service ui_selection=new User_UI_Selection_Service();

        //the logic

        return ui_selection;

    }

    public Store_Service Store() {

        Store_Service store=new Store_Service();

```

```

        // the logic
        return store;
    }

    public View_Service view() {
        View_Service view=new View_Service();
        // the logic
        return view;
    }

    public Update_Service Update() {
        Update_Service update=new Update_Service();
        // the logic
        return update;
    }
}

package Access;

public class Delete_Service {

}

```

**Figure 17 New Service Inseration in Access Component**

## CHAPTER SIX

### 6 CONCLUSION AND RECOMMENDATION

In the previous chapters, the main research problem and relevant sub-problems are solved for the purpose of achieving the main goal of this research (i.e. developing a framework for service component systems development). This chapter summarizes this study, draws major conclusions, surveys the achievements of this study, provides recommendations for prospective researchers, and indicates the need for future research.

#### 6.1 Conclusion

By investigating the requirements of current business-level IT systems, and comparing the advantages and disadvantages of the existing component-based and service oriented composition frameworks, this study provides a framework called SOCBD, with the goal of compensating for the limitations of current business system development paradigms.

SOCBD is a framework that combines the modification and manageability of CBSD and the flexibility of SOA. Modifiable SOCBD-based components provide any time modification. The SOCBD framework offers an alternative to older system development approaches, and provides a new solution which establishes a relationship between services and components for current business systems.

The implementation of an SOCBD for data exchange system or interoperability of organization does not conflict with existing technologies and system paradigms; hence the SOCBD framework could easily be accepted by the software engineering for organizational interoperability. By providing loosely coupled software mechanism for SOCBD-based components, which created by adopting various technologies, an SOCBD framework can rapidly accommodate the diverse requirements of future business.

Providing web services based data exchange system have two key advantages Platform/Language independent– Because from the users point of view all that is exchanged is XML which is a text based data format. The machinery that creates this XML behind the scenes can be on any platform using any language. The ubiquity of the HTTP protocol makes it the ideal

choice for integration. Firewall friendly because they work over HTTP web service traffic. This means that a System Admin does not have any additional firewall management headaches with extra ports being left open on a machine.

## **6.2 Recommendation**

How to develop a flexible enterprise system is a timely and crucial issue for today's business which use IT system as supportive for service deliverability to their business partners. However, there are so many relevant problems that need to be investigated carefully. We recommended as researcher to organizations or business whose aim to have Web based IT system for their service deliverability to full implement our service oriented architecture component based framework.

## Reference

- [1] Brown, A.W. and Wallnau, K.C., 1998. The current state of CBSE. *IEEE software*, 15(5), p.37.
- [2] Douglas, K.B., 2003. Web services and service-oriented architectures: the savvy manager's guide.
- [3] Stojanović, Z. and Dahanayake, A. eds., 2005. Service-oriented software system engineering: challenges and practices. IGI Global.
- [4] Newcomer, E., 2002. Understanding Web Services: XML, Wsdl, Soap, and UDDI. Addison-Wesley Professional.
- [5] Curbera, F., Golan, Y., Klein, J., Leymann, F. and Weerawarana, S., 2003. Business process execution language for web services, version 1.1.
- [6] Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Struble, S., Takacs-Nagy, P. and Trickovic, I., 2002. Web service choreography interface (WSCI) 1.0. *Standards proposal by BEA Systems, Intalio, SAP, and Sun Microsystems*.
- [7] Kotonya, G., Hutchinson, J. and Bloin, B., 2007. A method for formulating and architecting component and service-oriented systems. Stojanovic, Z. et al.(Hrsg.), pp.155-181.
- [8] Bussler, C., 2013. B2B integration: Concepts and architecture. Springer Science & Business Media..
- [9] Zenmyo, T., Kobayashi, T. and Saeki, M., 2004, September. Automated support for framework selection and customization. In *Automated Software Engineering, 2004. Proceedings. 19th International Conference on* (pp. 68-77). IEEE.
- [10] Bashir, M.S. and Qureshi, M.R.J., 2012. Hybrid Software Development Approach For Small To Medium Scale Projects: Rup, Xp & Scrum. *Cell*, 966, p.536474921.
- [11] Rasool, G., Aftab, S., Hussain, S. and Streitferdt, D., 2013. eXRUP: A Hybrid Software Development Model for Small to Medium Scale Projects. *Journal of Software Engineering and Applications*, 6(9), p.446.
- [12] Breivold, H.P. and Larsson, M., 2007, August. Component-based and service-oriented software engineering: Key concepts and principles. In *33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007)* (pp. 13-20). IEEE.
- [13] Causevic, A. and Vulgarakis, A., 2009, July. Towards a unified behavioral model for component-based and service-oriented systems. In *2009 33rd Annual IEEE International Computer Software and Applications Conference (Vol. 2, pp. 497-503)*. IEEE.

- [14] Mašek, K., Hnetyňka, P. and Bureš, T., 2009, August. Bridging the component-based and service-oriented worlds. In 2009 35th Euromicro Conference on Software Engineering and Advanced Applications (pp. 47-54). IEEE.
- [15] Wang, G. and Fung, C.K., 2004, January. Architecture paradigms and their influences and impacts on component-based software systems. In System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on (pp. 10-pp). IEEE.
- [16] Cervantes, H. and Hall, R.S., 2004, May. Autonomous adaptation to dynamic availability using a service-oriented component model. In Proceedings of the 26th International Conference on Software Engineering (pp. 614-623). IEEE Computer Society.
- [17] Jackson, M., 2012. Simplicity and complexity in programs and systems. In Conquering Complexity (pp. 49-72). Springer London.
- [18] Seinturier, L., Merle, P., Fournier, D., Dolet, N., Schiavoni, V. and Stefani, J.B., 2009, September. Reconfigurable sca applications with the frascati platform. In Services Computing, 2009. SCC'09. IEEE International Conference on (pp. 268-275). IEEE.
- [19] B. Khadidja, B. Djamel and H. Walid, 2014 Reconciling Component Based & Service Oriented Software Engineering: Application to e-Health System. ACIT International Conference on Information Technology. ACIT.
- [20] Olson, A.M., Raje, R.R., Bryant, B.R., Burt, C.C. and Auguston, M., 2005. UniFrame: a unified framework for developing service-oriented, component-based, distributed software systems. Service-oriented software system engineering: Challenges and practices, pp.68-87.
- [21] A. Tekalegn and F. Abera, 2012. Achieving Organizational Interoperability for Agricultural Research with Cross Organizational Business Process Modeling, HiLCoE Journal of Computer Science and Technology, Vol. 1, No. 1, pp 37-44.
- [22] A. Zewge and M. Belachew, Weldemariam et al. 2012. Designing an Architecture for Delivering Distributed Agricultural Information Services for Developing Countries. International Conference on ICT for Africa.
- [23] M. Broy, N. Diernhofer, J. Grünbauer, M. Meisinger, M. Rappl, S. Rittmann, B. Schätz, M. Schoenmakers, and B. Spanfelner, 2006. Service-Oriented Development - Whitepaper. Whitepaper, Technische Universität München.
- [24] I. Crnkovic and M. Larsson, 2002. Building Reliable Component-Based Software Systems. Artech House publisher.
- [25] C. Szyperski, S. Murer, and D. Gruntz, 2002. Component Software: Beyond Object-Oriented Programming, 2nd edition, Addison-Wesley.

- [26] S. Hashimi, 2003. Service-Oriented Architecture Explained, <http://www.ondotnet.com/>.
- [27] Z. Mahmood, 2007, The Promise and Limitations of Service Oriented Architecture International Journal of Computers, Issue 3, Volume 1, pp : 499-500
- [28] Heffner R, Leganza G (2011) SOA Adoption 2010: Still Important, Still Strong.
- [29] J. W. Creswell, 2009. Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, 3rd ed. Sage Publication, Inc.,
- [30] H. Edison<sup>a</sup>, N. B. Ali , R. Torkara, 2013 Towards innovation measurement in software industry, *The Journal of systems and Software*, vol. 86, no. 5, pp. 1390–1407.
- [31] IEEE Std 1471-2000, 2000. IEEE Recommended Practice for Architectural Description of Software Intensive Systems, <http://standards.ieee.org/catalog/software4.html#1471-200>
- [32] Bass L, Clements P & Kazman R. 2003. Software architecture in practice. 2nd ed. Reading: Addison Wesley.
- [33] C. Szyperski. 2002. Component Software – Beyond Object-Oriented Programming, Addison-Wesley.
- [34] <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>
- [35] C. Szyperski, S. Murer, and D. Gruntz, 2002. Component Software: Beyond Object-Oriented Programming, 2nd edition, Addison-Wesley,
- [36] T. Bures, P. Hnetynka, and F. Plasil, September, 2007. Runtime Concepts of Hierarchical Software Components, International Journal of Computer & Information Science, Vol. 8, No.5.
- [37] Fractal, <http://fractal.ow2.org/>
- [38] R. van Ommering, F. van der Linden, and J. Kramer. March 2000 The koala component model for consumer electronics software, In IEEE Computer, pp 78–85. IEEE,
- [39] Arcticus Systems, Rubus component model, <http://www.arcticus-systems.com>
- [40] M. Winter, C. Zeidler, C. Stich, ICSR 7 2002. The PECOS Software Process, Workshop on Components-based Software Development Processes.
- [41] A. Kaur and K. S. Mann, 2010 Component Selection for Component-Based Software Engineering, International Journal of Computer Applications, vol. 2, no. 1, pp. 109-114.
- [42] A. Vescan, 2008. Pareto Dominance - Based Approach for the Component Selection Problem, Second UKSIM European Symposium on Compute, pp. 58-63.

- [43] N. Haghpanah, S. Moaven, J. Habibi, M. Kargar, and S. H. Yeganeh, 2007. Approximation Algorithms for Software Component Selection Problem, in Proc. Asia Pacific Software Engineering Conference, pp. 159–166.
- [44] A. Kumar, P. Tomar, N. S. Gill, and D. Panwar, 2010. New Optimal Process for Selection of Software Components, in Proc. 1<sup>st</sup> National Conference on Next Generation Computing and Information Security, jointly organized by Computer Society of India and IMS, Noida, U.P., India, pp. 376.
- [45] Ning, J. Q. , 1996. A Component-Based Software Development Model. Proc. 20th Conf. Computer Software and Applications, Seoul, Korea, pp. 389-394.
- [46] Richard, M. A. ,1995. Emerging Standards for Component Software. Computer, Vol. 28(3), pp. 68-77.
- [47] Rizwan J. 2006. Reuse and Component Based Development. Proc. Int. Conf. Software Engineering Research & Practice, Las Vegas, USA, 146-150.
- [48] Hongyu Pei Breivold, Magnus Larsson, 2007. Component-Based and Service-Oriented Software Engineering: Key Concepts and Principles, 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO 2007), pp.13-20..
- [49] Web Services, <http://www.w3.org/2002/ws/>
- [50] The OSGi Alliance, Apr 2007. OSGi Service Platform Core Spec. R4, Version 4.1.
- [51] Wiki (accessed 14.05.2015). Various Enterprise service bus URL: [http://en.wikipedia.org/wiki/enterprise\\_service\\_bus](http://en.wikipedia.org/wiki/enterprise_service_bus).
- [52] M. Lian, March 2012. Introduction to Service Oriented Architecture, CSCI-5828 Foundations of Software Engineering.
- [53] Alejandro Sanchez, Rilwan Basanya, Tomasz Janowski, and Adegboyega Ojo, Enterprise Architectures - Enabling Interoperability between Organizations.
- [54] Abbas Rajabifar, Critical Issues in Global Geographic Information Management with a Detailed Focused on Data Integration and Interoperability of Systems and Data.
- [56] Strohmaier, M. B., 2005. PhD Thesis B-KIDE : A Framework and a Tool for Business Process Oriented Knowledge Infrastructure Development Knowledge Management.
- [57] National Enterprise Architecture Framework for Ethiopia (NEAFE).
- [58] Steinberger, G., M. Rothmund, and H. Auernhammer. 2009. Mobile farm equipment as a data source in an agricultural service architecture. Computers and Electronics in Agriculture, 65: pp. 238-246.

- [59] H. Holster, B. Fusai, G. Giannerini, S. Horakova, D. Martini, F. Teye and B. Ipema Current situation on data exchange in agriculture in the EU27 & Switzerland.
- [60] Schmitz, M., D. Martini, M. Kunisch, and H.-J. Möisinger. 2009a. agroXML Enabling Standardized, Platform-Independent Internet Data Exchange in Farm Management Information Systems, In: Sicilia, M.-A., Lytras, M. (Eds.), Metadata and Semantics. Springer US, pp. 463-468.
- [61] Murakami, E., A.M. Saraiva, L.C.M. Ribeiro Junior, C.E. Cugnasca, A.R. Hirakawa, and P.L.P. Correa. 2007. An infrastructure for the development of distributed service-oriented information systems for precision agriculture. Computers and Electronics in Agriculture, pp. 58:37-48.
- [62] Kaivosoja, J., M. Jackenkroll, R. Linkolehto, M. Weis, and R. Gerhards. 2014. Automatic control of farming operations based on spatial web services. Computers and Electronics in Agriculture, 100: pp. 110-115.
- [63] Nash, E., R. Nikkilä, L. Pesonen, K. Oetzel, W. Mayer, I. Seilonen, J. Kaivosoja, R. Bill, S. Fountas, and C. Sørensen. 2009b. Deliverable 4.1.1. Machine Readable Encoding for Definitions of Agricultural Crop Production and Farm Management Standards. EU, FutureFarm.
- [64] Robert A. Stegwee and Boriana D. Rukanova Identification of Different Types of Standards For Domain-Specific Interoperability.
- [65] Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F., 2006. Service-oriented computing research roadmap.
- [66] Papazoglou, M.P., Yang, J., 2002. Design Methodology for Web Services and Business Processes. In: Proceedings of the Third International Workshop on Technologies for E-Services (TES 2002). Volume LNCS; Vol. 2444., Springer-Verlag, pp54-64
- [67] Zirpins, C., Baier, T., Lamersdorf, W., 2003. A Blueprint of Service Engineering. In: First European Workshop on Object Orientation and Web Service (EOOWS), Darmstadt, Germany.
- [68] Tsai, W.T., 2005. Service-oriented system engineering: A new paradigm. In: Service Oriented System Engineering, SOSE . IEEE International Workshop, Beijing, China, pp 3- 6
- [69] Tsai, W.T., Wei, X., Paul, R., Chung, J.Y., Huang, Q., Chen, Y., 2007. Service-oriented system engineering (SOSE) and its applications to embedded system development. Service Oriented Computing and Applications pp. 3–17
- [70] K. Masek, P. Hnetynka and T. Bures Bridging the component-based and service-oriented worlds

- [71] Chakravarti, B., Ieee, S. M., and Varma, V., An Enterprise Architecture Framework for Building Service Oriented e-Governance Portal Architecture.
- [72] Sixth, T, Conference, I, & Milano, M., 2008. Business Process Management. Information Systems Research.
- [73] Kambhampaty, S., Chandra, S., Towers, T. S. R., and Road, R. B., 2006. Service Oriented Architecture for Enterprise Applications Architecture, pp. 48-54.
- [74] H.Kubicek et al., 2011. Organizational Interoperability in E-Government, DOI 10.1007/978-3-642- 22502-4\_7, Springer-Verlag Berlin Heidelberg.
- [75] IEEE Standard on interoperability, retrieved from <http://www.ieee.org/def?/definition.htm>.
- [76] GCI.,2006. 2016: The Future Value Chain, Global Commerce Initiative / Capgemini / Intel
- [77] H. Holster1, et al. Current situation on data exchange in agriculture in the EU27 & Switzerland
- [78] <http://www.fluxcorp.com/>
- [79] ASCX12, What is EDI?, <http://www.x12.org/x12org/about/>
- [80] M. Autili , Service-Orientated Component-Based Model, <http://www.di.univaq.it/marco.autili>
- [81] M. Desertot, H. Cervantes and D. Donsez, 2006 . FROGi: Fractal components deployment over OSGi, Software Composition, volume 4089 of Lecture Notes in Computer Science, pp. 275–290. Springer.
- [82] H. Cervantes, and R. S. Hall., Beanome, 2002. A Component Model for the OSGi Framework, Proceedings of the workshop Software Infrastructures for Component Based Applications on Consumer Devices, Lausanne.
- [83] L. Seinturier, P. Merle, D. Fournier, N. Dolet, V. Schiavoni, and J-B. Stefani., 2009. Reconfigurable SCA Applications with the FraSCati platform. In SCC '09: Proceedings of the 2009 IEEE International Conference on Services Computing, pp. 268–275, Washington, DC, USA.
- [84] B. Khadidja et al, 2014. Reconciling Component Based & Service Oriented Software Engineering: Application to e-Health System.
- [85] A. Tekalegn and F. Abera, 2012. Achieving Organizational Interoperability for Agricultural Research with Cross Organizational Business Process Modeling, HiLCoE Journal of Computer Science and Technology, Vol. 1, No. 1,

- [86] J. Y. Kim , C. G. Lee, S. H. Baek and J.Y. Rhee, June, 2015. Open farm information system data-exchange platform for interaction with agricultural information systems, AgricEngInt: CIGR Journal Open access at <http://www.cigrjournal.org>, Vol. 17, No. 2.
- [87] R. A. Stegwee and B. D. Rukanova Identification of Different Types of Standards for Domain-Specific Interoperability.
- [88] Minoli D. 2008. Enterprise architecture A to Z: frameworks, business process modeling, SOA, and infrastructure technology. Boca Raton: Auerbach Publications. Monson-Haefel R. 2009. 97 things every software architect should know: collective wisdom from the experts. Sebastopol: O'Reilly Media.
- [89] Bass L, Clements P & Kazman R., 2003. Software architecture in practice. 2nd ed. Reading: Addison Wesley.
- [90] Cervantes, H. and Hall, R.S., 2004, May. Autonomous adaptation to dynamic availability using a service-oriented component model. In *Proceedings of the 26th International Conference on Software Engineering* (pp. 614-623). IEEE Computer Society.
- [91] Limin S., Shangping R. Analysis and measurement of software flexibility based on flexible points. Published in the Proceedings of smef2006.

## **Appendix 1**

### ***Current System Challenges of the Organization***

According to interview and questioner undertook with the institutions to know current system challenges of the institutes the stakeholders answered the following:

*IT manger of ATA* answered the question what your institution current challenges according data exchange? According data exchange we only exchange through email and we have the data collected under our institution but there is no infrastructure to disseminate our data to others organizations. *IT manager of EIAR* answered the question what your institution current challenges according data exchange? According data exchange we use email and we exchange different data type with different organization. *ICT Director of MoA* answered the question what your institution current challenges according data exchange? According data exchange we use email with outsiders and system developed by our developers for internal but the problem is email is sufficient and the internal system is not flexible while there is change in requirement.

### ***The Requirements of the Organizations***

According to interview and questioner undertook with the institutions to know current system challenges of the institutes the stakeholders answered as described in section 4.1. Also the respondents or stakeholders answered the question what would you suggest or want the data exchanging system to be?

*IT manger of ATA* answered if we have infrastructure and data exchanging system we can disseminate our data to others also we will get data from them. *IT manager of EIAR* answered if we need data exchange system that responds to different data users on different data types. *ICT Director of MoA* answered if we have data exchange system that flexible to add and remove prerequisite for accessing the data for internal and external user.