Islamic University of Gaza
Deanery of Higher Studies
Faculty of Information Technology
Information Technology Program

A SOA Based Framework for the Palestinian e-Government Integrated Central Database

Prepared By:

Suhail M. Madoukh

Supervised By:

Dr. Rebhi S. Baraka

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology

1432/2011

بسم الله الرحمن الرحيم

الجامعة الإسلامية – غزة
The Islamic University - Gaza

عمادة الدراسات العليا
هاتف داخلي: 1150

ج س غ/35/
الرقم Ref 2011/11/01
التاريخ Date .....

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة عمادة الدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **سهيل محمد نمر مدوخ** لنيل درجة الماجستير في **كلية *تكنولوجيا المعلومات*** **برنامج تكنولوجيا المعلومات** وموضوعها:

A SOA Based Framework for the Palestinian E-Government Integrated Central Database

وبعد المناقشة التي تمت اليوم الثلاثاء 05 ذو الحجّة 1432هـ، الموافق 2011/11/01م الساعة التاسعة صباحًا، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. ربحي سليمان بركة	مشرفاً ورئيساً	.....
د. توفيق سليمان برهوم	مناقشاً داخلياً	.....
د. أشرف محمد العطّار	مناقشاً داخلياً	.....

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في **كلية *تكنولوجيا المعلومات*/ برنامج تكنولوجيا المعلومات.**

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق،،،

عميد الدراسات العليا

أ.د. فؤاد علي العاجز

P.O. Box 108, Rimal, Gaza, Palestine fax: +970 (8) 286 0800 فاكس Tel: +970 (8) 286 0700 ص.ب. 108 الرمال، غزة، فلسطين هاتف
public@iugaza.edu.ps www.iugaza.edu.ps

Acknowledgment

First of all, I thank Allah for giving me the strength and ability to complete this study.
Many thanks and sincere gratefulness goes to my supervisor Dr. Rebhi Baraka, without his help, guidance, and continuous follow-up; this research would never have been.
Also I would like to extend my thanks to the academic staff of the Faculty of Information Technology who helped me during my Master's study and taught me different courses.
I cannot forget to express my thanks to the members of the Governmental Data Integration Committee and the database administrator and software development staff at the governmental institutes.
I would like to thank my colleagues and classmates for making my study a great experience, useful, enjoyable, and full of warm atmosphere.
Last but not least, I am greatly indebted to my family for their support during my course studies and during my thesis work.

Abstract

The Integrated Central Database is one of the core components in the Palestinian e-Government Technical Framework. The current Integrated Central Database model lacks features such as: Interoperability, Flexibility, and Manageability. The purpose of this research is to propose a SOA based solution for the Central Database that achieves the above features.

This research presents and analyses the current architecture and implementation of the Palestinian e-Government Central Database model. The research proposes changing the current model into a Service Oriented Architecture (SOA) framework that is realized using Enterprise Service Bus (ESB) and Web Services. The proposed framework offers database replication and connectivity functionalities for central database.

The proposed framework is evaluated by using a scenario based software architecture evaluation method and proves that it achieves the quality attributes set as goals for the framework which are: Interoperability, Flexibility and Manageability. Moreover, a prototype of the framework is implemented and validates the framework correctness. A specific usage scenario for the framework is discussed and further proves that the framework accomplishes its functionality and quality attributes.

Keywords e-Government, SOA, ESB, Central Database, Database Replication, Web Services

عنوان البحث

إطار لقاعدة البيانات المتكاملة المركزية في الحكومة الإلكترونية الفلسطينية يعتمد على المعمارية الموجهة نحو الخدمة

الملخص

تعتبر قاعدة البيانات المتكاملة المركزية جزءا رئيسيا من مكونات الإطار الفني للحكومة الإلكترونية الفلسطينية. إن نموذج قاعدة البيانات الحالي يفتقر إلى صفات الجودة التالية: التوافقية بين الأنظمة، قابلية الإدارة، والمرونة، إن الهدف من هذا البحث هو تقديم مقترح لبناء قاعدة بيانات متكاملة مركزية تعتمد على مفهوم المعمارية الموجهة نحو الخدمة Service-Oriented Architecture (SOA) وتحقق الصفات المذكورة أعلاه.

يبدأ البحث بتقديم وتحليل قاعدة البيانات الحالية في الحكومة الإلكترونية الفلسطينية وكذلك كيفية تطبيقها، ويقترح البحث بأن يتم تحويل النموذج القائم لقاعدة البيانات إلى نموذج يعتمد على مفهوم SOA ويتم تحقيق هذا الأمر من خلال استخدام مفهوم "حافلة الخدمة للمؤسسة" Enterprise Service Bus (ESB) وخدمات الوب Web Services، إن الإطار المقترح لقاعدة البيانات سيقدم بشكل أساسي وظيفة الوصول إلى البيانات وكذلك دمج البيانات بين قواعد البيانات المختلفة ضمن قاعدة البيانات التكاملية المركزية.

تم تقييم الإطار المقترح من خلال منهجية "تقييم معمارية أنظمة البرمجيات بالاعتماد على السيناريوهات"، وتبين أنه يوفر الخصائص المطلوبة والتي تم تحديدها وهي التوافقية، قابلية الإدارة، والمرونة ، كما تم تطوير نظام برمجي لجانب من المقترح وذلك ليتم التحقق من مفهوم المقترح، وتم فحصه من خلال سيناريو استخدام محدد ومن خلاله تم التأكد من أن المفهوم والإطار المقترح قابل للتطبيق ويحقق الخصائص المطلوبة من حيث الوظائف الرئيسية وصفات الجودة.

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Glossaries

ATAM Architecture Tradeoff Analysis Method is a Scenario Based Software Architecture Evaluation Method.

BPEL Business Process Execution Language is an OASIS standard executable language for specifying actions within business processes with Web Services. Processes in BPEL export and import information by using Web Services interfaces exclusively.

CORBA Common Object Request Broker Architecture (CORBA) enables separate pieces of software written in different languages and running on different computers to work with each other like a single application or set of services.

CXF Celtix and XFire: Apache CXF is an open-source, fully featured Web services framework. It originated as the combination of two open-source projects: Celtix developed by IONA Technologies and XFire developed by a team hosted at Codehaus.

DCOM Distributed Component Object Model is a Microsoft technology for software distributed across several networked computers to communicate with each other.

EAI Enterprise Application Integration is software and architectural principles that allow for the integration of applications. EAI attempts to provide real-time access to data and processes with minimal changes to the existing applications and their underlying data structures.

EJB Enterprise Java Beans is a managed, server-side component architecture for modular construction of enterprise applications. The EJB specification is one of several Java APIs in the Java EE specification.

ESB	Enterprise Service Bus (ESB) is a software architecture model used for designing and implementing the interaction and communication between mutually interacting software applications in Service Oriented Architecture.
ETL	Extract, Transform, and Load (ETL) is a process in database usage and especially in data warehousing that involves: Extracting data from outside sources, Transforming it to fit operational needs (which can include quality levels), Loading it into the end target (database or data warehouse).
FTP	File Transfer Protocol (FTP) is a standard network protocol used to transfer files from one host to another host over a TCP-based network, such as the Internet.
HTTP	Hyper Text Transfer Protocol (HTTP) is a networking protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.
Hub-and-Spoke	An abstract software pattern used to transfer data between multiple systems. In contrast to the bus pattern, it uses a central component that coordinates all communication between senders and receivers.
J2EE	Java 2 Enterprise Edition (J2EE) is a widely used platform for server programming in the Java programming language. The Java platform (Enterprise Edition) differs from the Java Standard Edition Platform (Java SE) in that it adds libraries which provide functionality to deploy fault-tolerant, distributed, multi-tier Java software, based largely on modular components running on an application server.
JAX-WS	Java API For XML-based Web Services (JAX-WS) is a Java programming language API for creating Web Services. It is part of the Java EE platform from Sun Microsystems.
JBI	Java Business Integration (JBI) is a specification developed under the Java Community Process (JCP) for an approach to implementing a service-oriented architecture (SOA).

JDBC Java Database Connectivity (JDBC) is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases.

JMS Java Message Service (JMS) is a Java Message Oriented Middleware (MOM) API for sending messages between two or more clients. JMS is a part of the Java Platform, Enterprise Edition. It is a messaging standard that allows application components based on the Java Enterprise Edition (JEE) to create, send, receive, and read messages. It allows the communication between different components of a distributed application to be loosely coupled, reliable, and asynchronous.

JSF Java Server Faces (JSF) is a Java-based Web application framework intended to simplify development integration of web-based user interfaces.

JSP Java Server Pages (JSP) is a Java technology that helps software developers serve dynamically generated web pages based on HTML, XML, or other document types. JSP was designed to address the perception that the Java programming environment didn't provide developers with enough support for the Web.

MOM Message-Oriented Middleware (MOM) is software or hardware infrastructure supporting sending and receiving messages between distributed systems. MOM allows application modules to be distributed over heterogeneous platforms and reduces the complexity of developing applications that span multiple operating systems and network protocol.

QoS Quality of Service (QoS) is sometimes used as a quality measure; it refers to the level of quality of service, i.e. the guaranteed service quality.

REST Representational State Transfer (REST) is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

RPC Remote Procedure Call (RPC) is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction.

SLA Service Level Agreement (SLA) is a part of a service contract where the level of service is formally defined. The term SLA is sometimes used to refer to the contracted delivery time or performance.

SOA Service-Oriented Architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services. These services are well-defined business functionalities that are built as software components (discrete pieces of code and/or data structures) that can be reused for different purposes.

SOAP Simple Object Access Protocol (SOAP) is an XML-based protocol for exchange of information in a decentralized, distributed environment.

TCP Transmission Control Protocol (TCP) is one of the core protocols of the Internet Protocol Suite. TCP is one of the two original components of the suite, complementing the Internet Protocol (IP), and therefore the entire suite is commonly referred to as TCP/IP. TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer.

UDDI Universal Description Discovery and Integration (UDDI) is a platform-independent, XML-based registry for businesses all over the world to be listed on the Internet; it is an open industry initiative (sponsored by OASIS) enabling businesses to discover each other and define how to interact over the Internet.

UDP User Datagram Protocol (UDP) is one of the core members of the Internet Protocol Suite, the set of network protocols used for the Internet. With UDP, computer applications can send messages, in this case referred to as datagram, to other hosts on an Internet Protocol (IP) network without requiring prior communications to set up special transmission channels or data paths.

W3C World Wide Web Consortium (WWW) was created in 1994 for leading the World Wide Web to its full potential by developing common protocols which promote its evolution and ensure its interoperability.

WSDL Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages which contain Document-oriented or procedure-oriented information.

WS-I Web Service Interoperability (WS-I) is an open industry organization that promotes Web services interoperability.

XML Extensible Markup Language (XML) is a Meta-Language written in Standard Generalized Markup Language (SGML) that allows using to allow for easy interchange of documents on the World Wide Web.

XSD XML Schema Definition (XSD) is a recommendation of the World Wide Web Consortium (W3C), specifies how to formally describe the elements in an Extensible Markup Language (XML) document. This description can be used to verify that each item of content in a document adheres to the description of the element in which the content is to be placed.

List of Appendices

Chapter 1

Introduction

This chapter introduces the thesis by describing the Palestinian e-Government and the Integrated Central Database, the thesis problem, the research objectives, the importance of the research, the scope and limitation of the thesis work, the methodology, resources and tools.

1.1. The Palestinian e-Government and Integrated Central Database

e-Government has drawn an increasing attention recently; it represents a promising initiative that makes people's life easier. e-Government can be defined as a way for an effective use of Information and Communication Technologies (ICT) in the government conduct in order to enhance government-citizen interaction [76]. The ultimate goal of e-Government is to improve government–citizen interactions through an infrastructure built around the "life experience" of citizens [42]. Many aspects of e-Government need to be addressed thoroughly, at both process and technical levels. In the year of 2005 the Palestinian Government adopted the e-Government initiative [47]. A Technical Framework for the e-Government initiative was specified and the Integrated Central Database, which is one of the main components in the framework, was implemented [52]. The Integrated Central Database integrates different data sources replicated from various ministries and consolidates them in one location. The Integrated Central Database relieves ministries from accessing different data sources by providing a single point of access to various data sources. The currently used Integrated Central Database architecture is modeled and realized to provide data integration between governmental institutes and to provide inter-ministry exchange of data. This database can be thought of as a broker that integrates data from different sources and then allows the sharing of data between various ministries.

The current Integrated Central Database architecture lacks Interoperability, Flexibility, and Manageability features. It is restricted to using a specific database type for the

replication which is based on Oracle DBMS, and connectivity to the database is limited to be from the Internal Government Network as well as over a specific transport port.

The Integrated Central Database has a limited management capability for the database integration process.

Distributed systems are considered as a solution to overcome the shortcomings of the Integrated Central Database. However, while Distributed systems integration models such as Common Object Request Broker Architecture (CORBA) and Distributed Component Object Model (DCOM) have been successfully implemented on many platforms, their limitation arises as enterprises management are not centralized, the limitations can be obviously seen when the solutions are built on those protocols that depend on a single vendor. These protocols closely depend on a single administered environment and hence not used between enterprises, which therefore leads to lack of interoperability [10].

To remedy the deficiencies of distributed systems, Service Oriented Architecture (SOA) is introduced. SOA is an architecture that uses the "service" as a basic construct in the development approach and builds on distributed, loosely coupled, and interoperable components of software [29][42]. SOA provides a solution to shared and distributed services development [38][51], and achieves high Interoperability, Flexibility, and standardization by utilizing the description, discovery, and invocation of services. SOA based integration helps in achieving qualities such as: Interoperability, Flexibility, and Manageability [54]. Moreover the concept of SOA was supported by various companies like IBM and Microsoft. They argue that in order for SOA to succeed it must be implemented on open standards [10]. Consequently, the shortcomings of the Integrated Central Database Architecture can be eliminated by transforming the current architecture into SOA-based, which can be realized using Web services and the Enterprise Service Bus (ESB) platform.

In this research we propose to use ESB and Web Services in the SOA realization process. The use of Web services in e-Government enables the governmental institutions to provide additional services by defining new services that emerge from other e-Government services [33][42]. To realize the concept of SOA model, one would use the Enterprise Service Bus [42][53][57]. ESB will act as the middleware glue

infrastructure that holds SOA together. It integrates and manages the communication between different Web services [51], applications, and data sources. ESB will provide functionalities such as routing and transporting service requests, security assurance, service orchestration, and management capabilities. The e-Government Integrated Central Database framework will be designed based on SOA approach that is realized through using Web services and ESB.

The work methodology of the research will lead to proposing a framework based on SOA for the Integrated Central Database of the Palestinian e-Government Technical Framework. The framework will be evaluated using a scenario based software architecture evaluation method. And the correctness of the framework will be validated through using a proof-of-concept methodology in which a prototype will be implemented.

1.2. Problem Statement

The increasing size of disparate, distributed and heterogeneous government databases and the high demand for the exchange of data between the different governmental institutes create an integration problem. The Integrated Central Database is one of the core components in the Palestinian e-Government Technical Framework. The currently implemented architecture of the database relies on replicating subsets of the government institutes databases into the Integrated Central Database. The main functionalities of the Integrated Central Database are the replication and accessibility. Both functionalities suffer from the lack of vital features which are: i*nteroperability, flexibility and manageability*. Problems of the current architecture appear when trying to replicate a government institute database with a different type database of the Integrated Central Database. The problem also appears when clients trying to access the Integrated Central Database over a transport, driver, or an Application Programming Interface (API) that is not natively supported by the Integrated Central Database. Another problem emerges from the inability to attain a central point of management for the operation of the Integrated Central Database.

The above shortcomings can be overcome by transforming the current architecture into a SOA-based one. Even though many researchers proposed SOA-based approaches for

the data exchange and integration issue, but their proposed solutions do not address all three problems mentioned above, and they are based on requirements that do not completely match our case.

The problem of this research is how to build a SOA-based framework for the Palestinian e-Government Integrated Central Database that achieves *interoperability, flexibility, and manageability*.

1.3. Objectives

In this section we present both main and specific objectives of the research work.

1.3.1 Main Objective

The main objective of this research is to build a SOA-based Framework for the Integrated Central Database of the Palestinian e-Government Technical Framework that achieves *interoperability*, *flexibility*, and *manageability* with emphasis to database functionalities related to replication, connectivity.

1.3.2 Specific Objectives

The specific objectives of this research are:

- Analyze the current Integrated Central Database Architecture to determine the shortcomings and requirements with respect to *interoperability*, *flexibility*, and *manageability* quality attributes.
- Build a framework to transfer the Integrated Central Database Architecture into a SOA based one.
- Evaluate the proposed framework for *interoperability*, *flexibility*, and *manageability* using scenario based software architecture evaluation method.
- Validate the framework using a proof-of-concept method through implementing and deploying a prototype of the framework.
- Present a usage scenario of the prototype to show the framework solution to show the quality attributes achievement.

1.4. Importance of the Research

- The proposed framework would have a great value towards the advancement of the e-Government initiative in Palestine, which would result in speeding the advancement of the e-Government services and hence citizens would be the ultimate beneficiary of it.
- The framework would also allow for an easy and standard access to the Integrated Central Database for the non-governmental institutes.
- The framework would allow for the easiness of adding ministries databases to the replication and access of the Integrated Central Database.
- The idea of the research idea has got an initial approval by the Ministry of Telecom and IT, the institute that oversees the e-Government initiative, and hence the research will be adopted and used by the concerned departments.
- The deployment of the framework would present a valuable chance for the IT staff at the ministries to enhance their experience and knowledge in SOA technology, and hence would move the current Web computing trends in the government sector towards SOA based.
- Other e-Government initiatives or related information systems integration projects can benefit from the framework even though their requirements may vary because the SOA principles enhance the reusability feature.

1.5. Scope and Limitations of the Research

- The scope of the research will address and achieve all functionalities provided by the current Integrated Central Database in the e-Government Technical Framework, in addition to achieving the *interoperability, flexibility,* and *manageability* quality attributes.
- Quality attributes such as scalability, fault-tolerance will not be addressed by the proposed framework, because of the time constraint for completing the thesis work.
- The framework will not be fully implemented, but rather a prototype of it with a specific usage scenario will be implemented. The prototype will provide a proof of concept for the proposed framework.

- Even though the framework will be designed to support different database types, the implementation of the replication service will consider only two database products which are Oracle and MySQL.
- The available replication tools that comes with a specific database will be used if applies, e.g. Oracle to Oracle replication, synchronization, and consistency checks are supported natively by the vendor, hence they will not be implemented, but the management of the replication as a service will be addressed.
- The security issue will be addressed in terms of authentication of users and their authorization to access services. Every service will be accessed based on specific usage permission and user identification mechanism. The security issue of messaging exchange and communication between different services will be handled by the transport, the network layers, and the ESB platform.
- Online help, or system documentation will not be part of the framework, since services are described by their interface files which are built based on standard approach.

1.6. Methodology

To accomplish the objectives of the research, the following methodology will be followed:

- Analyze the current Integrated Central Database architecture against the goals which are: *interoperability, flexibility, and manageability*.
- Study and investigate SOA approaches for integrating, sharing, and accessing distributed and central databases.
- Study alternatives of ESB platforms and decide on the suitable one to be used in the implementation of the framework.
- Develop the proposed framework:
  - Specify the requirements and the framework architecture.
  - Define the components of the framework.
  - Specify the interaction between framework components.
  - Define the accessibility approaches to the Integrated Centralized Database for the front-end clients.

- Address the ethical concerns related to the framework, this is because the Integrated Central Database includes huge repository of data including personal data, governmental institutes' data, and private data.
- Evaluate the framework for the defined quality attributes using a scenario based software architecture evaluation.
- Implement a prototype as a validation of the framework using the proof of concept. The prototype is to allow for a specific usage scenario, including the front-end access to the Integrated Central Database through a Web portal and Web services. The prototype is to include the following tasks:
  - Choose three informational services from different ministries to be implemented.
  - Decide on service realization strategy for different services, e.g. top-down, bottom-up, meet in the middle.
  - Develop an orchestration service for two of the services.
  - Implement the replication service for the support of Oracle to MySQL, and Oracle to Oracle replication.
  - Implement the front-end access Web interface that will invoke the implemented services.
- Verify the prototype goals achievements by presenting a specific usage scenario.

1.7. Resources, Methods, and Tools

The following resources and methods will be required:

- Working network platform including database server, Web server, and Web client.
- Database Managers in different ministries: For surveying the used database type and their replication options.
- Other research groups working in the same area: For advice.
- The required software to conduct this research will be software packages and development tools, and platform that support Web services and ESB framework. The tools are (Oracle database, Java Business Integration (JBI), MySQL database, Netbeans IDE, JSP, Apache, Tomcat, Linux)

- Software Architecture Evaluation Methods such as Architecture Tradeoff Analysis Method (ATAM) for framework evaluation, and proof-of-concept as a means for validating the framework, where a showcase application has been developed to prove the correctness of the implementation and the usefulness of the concepts.
- Services realization options such as top-down, bottom-up, and meet-in-the middle.

1.8. Thesis Structure

This thesis consists of eight mainly chapters: Introduction, Theoretical Foundation, Related Works, Evaluating the Current Palestinian e-Government Central Database Model, Framework Structure and Components, Framework Prototype, Framework Evaluation, and Conclusions and Future works. The main points discussed in the chapters are listed below:

- Chapter 1 Introduction: gives a short introduction about the Integrated Central Database and the thesis problem and objectives.
- Chapter 2 Technical Foundations: describes the technical foundations needed for thesis work, SOA, Web Services and BPEL, Replication, and Software Evaluation.
- Chapter 3 Related Works: presents related works to the thesis.
- Chapter 4 Evaluating the Current Palestinian e-Government Central Database Model: presents and analyze the current Integrated Database model, and discusses the shortcomings of the model.
- Chapter 5 SOA-based Framework: presents the proposed the SOA-based framework for the Integrated Central Database and describes the components and their interaction.
- Chapter 6 Framework Prototype: is devoted to the presenting the implementation of the framework prototype and describes prototype architecture and the implemented service.
- Chapter 7 Framework Evaluation: presents the evaluation of the framework using scenario based software architecture evaluation method, and validates the prototype using the proof of concept validation approach.
- Chapter 8 Conclusions and Future Work: discusses the final conclusions and presents possible future works.

Chapter 2

Technical Foundations

In this chapter the fundamental concepts and technical knowledge which represent the basis for understanding of the thesis work are presented. First the Service-Oriented Architecture (SOA) is introduced, followed by Web services, Business Process Execution Language (BPEL), Universal Description Discovery and Integration (UDDI), Enterprise Application Integration (EAI), Enterprise Service Bus (ESB), database replication, and finally evaluation methods for software architecture and proof-of-concept.

2.1. Service Oriented Architecture (SOA)

Service Oriented Architecture (SOA) is considered as an evolution of distributed computing and modular programming. It is a style of information system architecture that enables the creation of applications that are built by combining loosely coupled and interoperable services. SOA is a style of coarse grained, loosely coupled software architecture [37]. SOA also describes IT infrastructure which allows different applications to exchange data with one another as they participate in business processes [68]. SOA is an approach to IT that considers business processes as reusable components or services that are independent of applications and the computing platforms on which they run. SOA is not tied to a specific technology; it can be realized using different technologies, such as: RPC, DCOM, CORBA and Web Services.

Following we discuss the rationale behind using SOA, SOA architecture, SOA layers, and implementations.

2.1.1 SOA Benefits

Service-orientation aims at a loose coupling of services with operating systems, programming languages and other technologies which underlie applications [68]. Many businesses adopted the Service-Oriented Architecture due to the changing in business

demands regarding quality and flexible IT [66]. SOA allows software designers to design solutions such as assemblies of services; and hence SOA-based systems can be independent of the development technologies and platforms. In SOA environment independent services can be accessed without knowledge of their underlying platform implementation [67].

2.1.2 SOA Architecture

A typical architecture of SOA is depicted in Figure 2.1, which includes three main roles that interact using standard messaging. The roles are service provider, service registry and service client [53]. The service is first published by the service provider to the service registry, which is a repository that holds services interfacing information. The service client searches the service registry for a specific service, and gets its binding information. The client uses service binding information to consume the service provided by the service provider. The service registry in the architecture is optional and so the SOA solutions can be designed with service provider and client only.

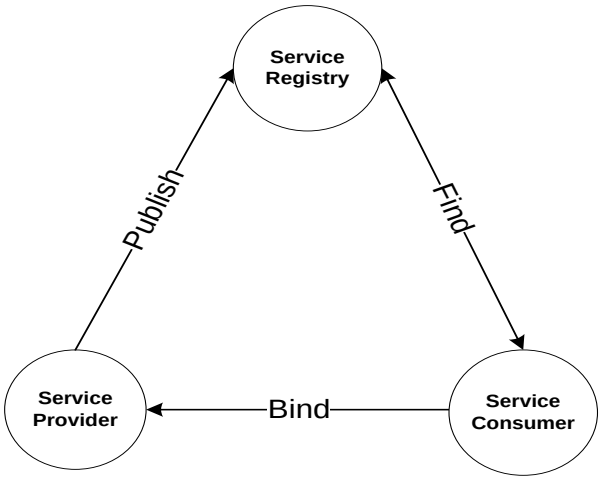

Figure 2.1: A Typical SOA Architecture

2.1.3 SOA Layers

SOA solutions are built using layers which provide a level of abstraction; layers show the conceptual structure of services in SOA even though there is no wide agreement in the literature regarding the name of the layers. Common layers names and their explanation is presented as follows [19]:

- Presentation Layer: This presentation layer contains the application front-ends and services. These provide access to the SOA, and provide communication between end users and the SOA. Access to services is provided for business-to-business situation, along with users within an organization.
- Business Process Layer: The business process layer contains services that contain a full business process. These services are composed with orchestration methods such as BPEL and the services of the services layer (more about BPEL in section 2.3).
- Services Layer: The services layer contains the services that provide access to internal applications and data. The services layer also contains proxies that can access services that other companies have in the presentation layer. Different service consumers use the services in this layer; therefore, they should be as generic as possible, without being redundant.
- Application Layer: The application layer contains the applications of an organization. Some sources also include the data stores in this layer. The applications provide the organization with some sort of functionality (e.g. keep all employee data in a database).

2.1.4 SOA Implementation

To realize a SOA solution, different technologies can be used such as CORBA, DCOM, RPC, or Web Services. Web services (discussed in section 2.2) are a suitable choice to build SOA solutions. SOA Infrastructure is realized through bus approach where point to point interaction between services and components without a use of a middleware, or based on a hub and spoke approach using an ESB. An ESB provides an implementation backbone for an SOA that treats applications as services. It establishes proper control of messaging as well as applies the needs of security, policy, reliability and accounting, in an SOA [53], ESB will be explained in section 2.6.

2.2. Web Services

Web services are a good technology to build Service Oriented Architectures. A Web Service is defined as a software system designed to support interoperable machine-to-machine interaction over a network [73]. Web services provide the basis for the

development and execution of business processes that are distributed over the network and available via standard interfaces and protocols [56]. Web services may use the Internet as the communication medium (as well as other transport protocols) and open Internet-based standards, such as the Simple Object Access Protocol (SOAP) as transmission medium, the Web Services Description Language (WSDL) for service definition and the Business Process Execution Language (BPEL) for orchestrating services. Web services are simple, easy to understand, requires inexpensive technology, no major investment needed, and can start small and proceed incrementally.

2.2.1 Simple Object Access Protocol (SOAP)

SOAP stands for Simple Object Access Protocol, which is an XML-based protocol for the communication in a Web Services environment. SOAP messages enable the communication between the service provider and requester. From the requestor's point of view, SOAP is needed for the discovery and invocation of a service. SOAP uses HTTP and XML to solve the problem of inter-acting the interfaces between the various platforms in a network, because they are both platform independent. SOAP explains what data should be in the http-header, as well as what data should be in an body of the soap environment of a HTTP message, so that an application in one side can call an application in another computer side and transfer information (data, etc.) between the two sides. Information needed by a Web server that works with the SOAP protocol is contained in a SOAP message, which consists of a SOAP Envelope, a SOAP header and a SOAP body. Figure 2.2 shows the overall structure of a SOAP message [39].

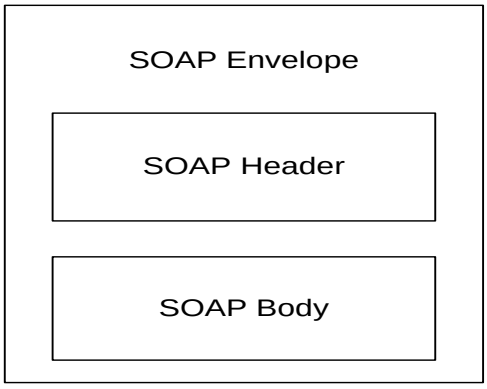

Figure 2.2: SOAP Message Structure

The envelope represents the entire package of data that is being used to explain how information should be transferred. As part of this envelope, the optional header can

contain information about the routing and the delivery options for the SOAP message. The mandatory body, also as a part of the envelope, contains the actual data, which is usually the method or operation one is invoking.

2.2.2 Web Services Description Language (WSDL)

Web service requesters need to know how to access Web services and get their description through a standard way, and to provide such facility for the description of the Web Services, WSDL is introduced. WSDL stands for Web Services Description Language. It is used to describe Web Services. Usually the service provider creates a description about the offered service during the development of a service. For a service requestor it is important to know where a service is located. Furthermore the requestor needs to know what kind of messages the services will understand and what kind of operations are offered, as well as how the messages are encoded and which protocol is used for exchanging the messages. This information can be provided to the requestor by the WSDL file of a service [77]. Figure 2.3 depicts WSDL document structure which is an XML file that mainly includes sections for the messages, operations, binding, and service location.

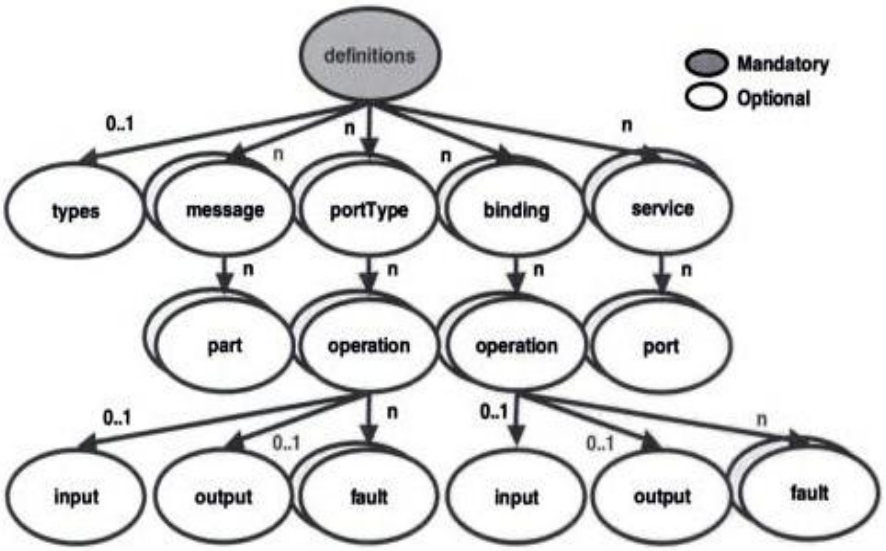

Figure 2.3: WSDL Document Structure [7]

2.3. Business Process Execution Language (BPEL)

As discussed in subsection 2.1.3 in SOA Layers, SOA can have a process layer to provide a composite process from various Web Services and to provide a way to realize processes. Business Process Execution Language (BPEL) is used for service orchestration. One standard for BPEL is Web Services Business Process Execution Language (WS-BPEL or BPEL4WS) which is maintained by Organization for the Advancement of Structured Information Standards (OASIS) [74]. In addition to service composition, BPEL can also be realized by means of virtualization tier and Web Services [71]. Business processes can be composed of different calls to specific Web Services. With WS-BPEL the order in which specific services are called can be controlled and this is referred to as service orchestration. WS-BPEL supports two types of business processes.

1- The executable processes which specify the exact details of business processes and are executed by a BPEL engine.
2- Abstract business process which specifies the public message exchange between the client and the service.

The benefit of WS-BPEL is that it allows business processes to be changed just by adjusting the WS-BPEL file. Figure 2.4 (from [60]) shows an example of a WS-BPEL process. Service A wants to consume a service B in the other enterprise. Data is sent between A and B is in the form of a SOAP document. Service B is a business process and uses service C and D. Service A does not care what happened behind service B just as long as it does what is should do. If service B wants to adjust the process by replacing service B with service E (which is not shown), they can do by changing the WS-BPEL file. Service A will not notice any change.

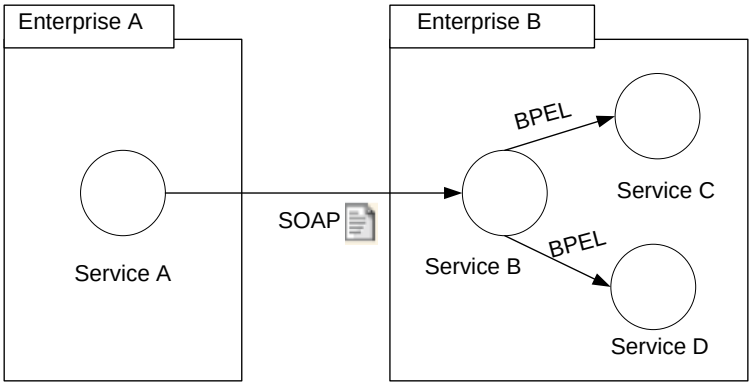

Figure 2.4: Service Orchestration [60]

2.4. Universal Description Discovery and Integration (UDDI)

One component of a typical SOA Architecture is the service registry, which is used to facilitate the finding of services by the requestor and to realize the service registry. To implement such functionality, the Universal Description Discovery and Integration (UDDI) initiative was presented by IBM, Microsoft and Ariba. It actually initiated from the e-business community. UDDI comes to solve the problem that challenges the requestor to discover and find a service to use. The UDDI registry can be thought as a centralized Web Services search engine helping the service consumer to find adequate service offerings [77]. In a UDDI registry consumers may find:

- Information about businesses and organizations offering Web Services.
- Descriptions of the Web Services that these organizations provide.
- Information about technical interfaces to these Web Services.

2.5. Enterprise Architecture Infrastructure (EAI)

To integrate application and data between enterprises, various integration ways are addressed. Message Oriented Middleware (MOM) is one of the traditional approaches for enterprise integration which is an infrastructure that uses asynchronous messaging to decouple applications from each others. It is based on a central message queue called message broker to which applications are connected through a centralized united interface. Figure 2.5 depicts a simplified architecture for the MOM.

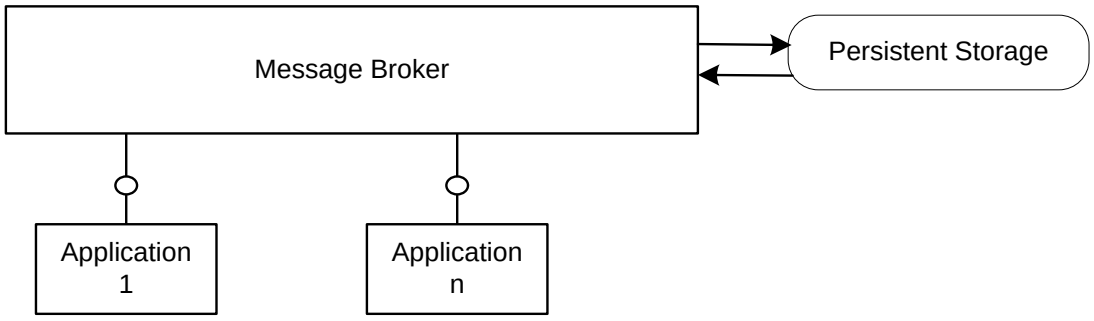

Figure 2.5: Simplified Architecture of Message Oriented Middleware [44]

As shown in Figure 2.5, the message broker has a persistent storage area for storing messages so that sender and receiver do not need to be connected at the same time, this leads to decoupling. Moreover, messages are routed by the message broker, this means to provide the ability to deliver a single message to multiple recipients and it allows for

message transformation where message can be formatted for different application format [44].

Even though MOM presents a solution for enterprise integration, it faces a problem of using proprietary protocols and platform specific interfaces and deployments. Such problem leads to having applications that are dependent on the infrastructure and causes Interoperability problem. In a typical enterprise such issue turns to having multiple MOM based infrastructures. Another problem with MOM is that it has popular hub and spoke Enterprise Architecture Infrastructure (EAI) platform, in which all integrated applications work through a single message broker, creates a single point of failure, which presents a high risk for a complex business system. To overcome the shortcomings of traditional enterprise architecture infrastructure the Enterprise Service Bus is introduced [18].

2.6. Enterprise Service Bus (ESB)

To realize a SOA solution an Enterprise Service Bus (ESB) can be used. An Enterprise Service Bus is an open standards, message based distributed integration infrastructure that provides routing, invocation and mediation services to facilitate the interactions of disparate distributed applications and services in a secure and reliable manner [44]. ESB is a critical infrastructure that should be implemented and designed based on architectural blueprint. So, an ESB is a product, which evolves from architecture. An ESB is valuable to the implementation of a service-oriented architecture (SOA) [12] and ESB provides a fundamental support for EAI. It provides a middleware for accessing and transforming information to several protocols such as Java Message Service (JMS), SOAP, HTTP, FTP and TCP. It allows the communication between these different protocols with the support of adaptors [63], where adaptors are used to connect applications to the ESB. To ensure Interoperability, the components of the ESB and the mechanism for connecting resources must be based on open standard. ESB is realized through using service containers distributed over the network. The containers host integration service such as routers and transformers, and provide services with communication facilities. Messaging infrastructure is built on top middleware systems which guarantee message delivery, such as JMS middleware. Figure 2.6 depicts a simplified general architecture view of an ESB. The ESB, as shown in Figure 2.6,

integrates a J2EE application using JMS, interfaces with legacy applications, and Web services. Moreover a distributed query engine is attached to the ESB which is normally based on XQuery or SQL. The query engine enables the creation of data services to abstract the complexity of underlying data sources. As shown in Figure 2.5, a main use for ESB is to act as the intermediary layer between a portal server and the backend data sources that the portal server needs to interact with [55].

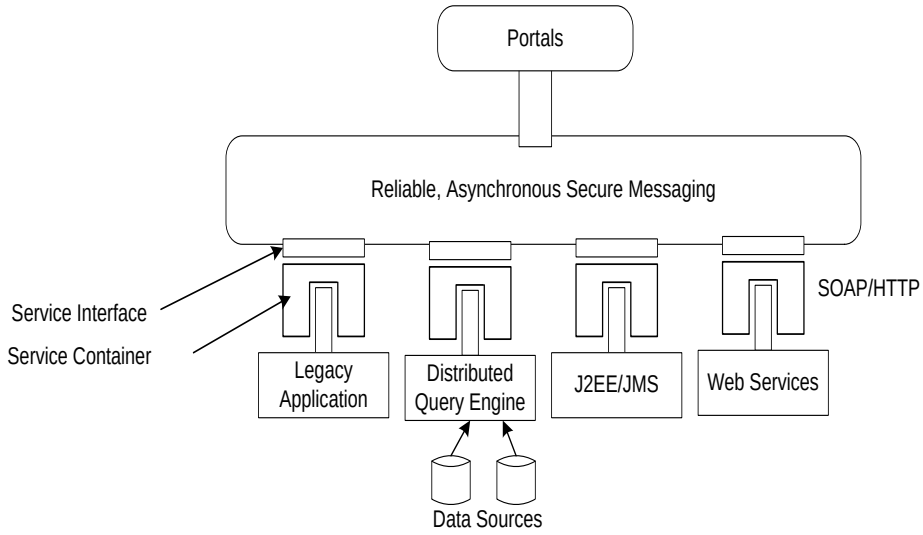

Figure 2.6: A Typical ESB Connecting Diverse Applications [55]

ESB should achieve a goal of providing interaction, messaging and integration without writing code. ESB should provide generic components which can be configured to realize a desired scenario. With its distributed deployment infrastructure, an ESB can efficiently provide central configuration, deployment, and management of services that are distributed across the extended enterprise [12]. In the following two sections ESB functions and features, and ESB implementations are presented.

2.6.1 ESB Functions and Features

ESB performs different functions such as: connectivity, routing, transformation, security, management, validation and processing of messages, orchestration services, and performing publish and subscribe functions between disparate and distributed applications. Below are listing of main functions that may be supported by different ESBs [44], [20], [63], [45], and [36].

- Invocation: Which is the service ability to send requests and receive responses from integration services and resources, and handles the underlying protocols such as

TCP, UDP, HTTP, SSL, and communication mechanisms could be JBI, RMI, JDBC, SMTP, FTP, or POP3.

- Routing: Which provides the ability to decide about the destination of a message during its transport, and routing allows decoupling the source of a message from the ultimate destination.
- Mediation: In which applications rarely agree on common data format, hence this feature is important for data transformation.
- Adaptors (connectors): In which ESB provides a whole range of application adaptor, such as Enterprise Resource Planning (ERP), Supply Chain Management (SCM), and Customer Relation Management (CRM). Using prefabricated adapters reduces the work required to integrate applications into a Service-Oriented Architecture, such as File/FTP adapter service, Database Adapter Service, and JMS Adapter Service, etc. A typical function of database adapter service can be inbound and outbound. An inbound database adapter service sends an XML message to e.g. an Enterprise Service Bus when a SQL insert, update, or delete operation is performed against a database. An outbound database adapter transforms the contents of an XML message into a SQL insert, update, or delete operation on the target database [26].
- Process Orchestration: Which provides an engine to execute business processes described with the Web Services Business Process Execution Language (WS-BPEL). This engine is controlled by the process description then coordinates the collaboration of the services connected to the bus.
- Complex Event Processing: ESB provides a way where an asynchronous message can be seen as an event especially when using publish-subscribe channel. An ESB may provide event interpretation, event pattern matching which enable event-driven architectures.
- Security: In which ESB provides secure messaging, to be able to encrypt and decrypt the content of messages. Handles authentication and access control for messaging endpoints and to use secure persistence mechanisms.
- Management: This provides a central mechanism for configuration and administration of the bus. It offers a unified management console for monitoring and administration of infrastructure and integration scenario. And provides audit

and logging facilities for monitoring infrastructure and integration scenario and possibly also for controlling process execution.

- Integration Tooling: where ESB provides graphical design-time tooling for professional development with an ESB. And a deployment and testing tool should be available.
- Quality of Service: In which transaction control and compensation is addressed, failover at service and service container, and provides distributed network topology with flexible deployment for performance and scalability.

2.6.2 ESB Implementation

Some ESB implementations are based on Java Business Integration (JBI), which defines an architecture that allows integration products to be built based on components that can be plugged into the JBI environment. JBI is based on two main concepts: the Service Engines (SEs) and the Binding Components (BCs). SE provides functionality to other JBI components and can consume services provided by other JBI components. And the BC allows the connectivity outside the JBI environment. The main benefit of it is that JBI component may be reusable among JBI-compliant ESB [20].

The majority of ESBs deal with Java Message Service (JMS) for managing the delivery and reception of messages between different components. Other used technologies in ESBs are generally REST, HTTP, JDBC, TCP, UDP or CXF [20].

There are several ESBs products, both commercial and open source [20]. Commercial products such as IBM Web Sphere Message Broker [23], TIBCO Business Work [70] or Oracle ESB [26], and open-source implementations such as ServiceMix [3], Fuse ESB [49], Mule [48], Open-ESB[50], JBoss ESB [27] or Petals [59].

A typical usage scenario for Open-ESB, which is built on top of Java Business Integration (JBI), shows the following components are used: JSP/JSF Web page, JAVA DB, JMS Queue, Composite Application BPEL, EJB-based Web service, and Servlet-Based Web Services [6].

2.7. Database Replication

The thesis problem presents the integration issue between various types of databases. To accomplish the integration of such databases to be centralized, a replication technique is used. Replication is a technique in distributed data sharing where a component is copied, or replicated, and kept consistent in order to improve availability and performance [65]. Replication is also defined the process of creation and maintenance of duplicate versions of database objects in a distributed database system [40]. Replication is used when there is a series of databases that should be and should remain identical, even when changes are made at one of the replicas. Replication algorithms process these changes and propagate them to the replicas. The replicas are updated consistently.

2.7.1 Replication Types

The replication tools may be selected based on the type of replication it supports [1]. The capabilities and performance characteristics varies from one type of replication to another. A replication strategy may be selected based on two basic characteristics: Where and When. When the data is updated at one site, the updates have to be propagated to the respective replicas. When the updates can be propagated can be achieved by Synchronous (eager) and Asynchronous (lazy) methods and where the updates can take place can be achieved by update everywhere and primary copy (master-slave) methods. Synchronous replication (Master-Slave replication) works on the principle of Two-Phase commit protocol [9]. In a two-phase commit protocol, when an update to the master database is requested, the master system connects to all other systems (slave databases), locks those databases at the record level and then updates them simultaneously. If one of the slaves is not available, the data may not be updated. The consistency of data is preserved; however it requires availability of all sites at the time of propagation of updates. There exist two variations of Asynchronous replication (Store and Forward replication) i.e. periodic and trigger-based. In Periodic replication, the updates to data items are done at specific intervals and in trigger-based replication the updates are propagated only when necessary (usually based on firing of event in a trigger). Various forms of replication strategies are as follows:

- Snapshot Replication*:* In snapshot replication, a snapshot or copy of data is taken from one server and moved to another server or to another database on the same server. After the initial synchronization, snapshot replication can refresh data in published tables periodically. Though snapshot replication is easiest form of replication, it requires copying all data items each time a table is refreshed.

- Transactional Replication: In transactional replication, the replication agent monitors the server for changes to the database and transmits those changes to the other backup servers [40]. This transmission can take place immediately or on periodic basis. Transactional Replication is used for server-server scenarios. Materialized views are classified as a transactional replication.

- Merge Replication: Merge replication allows the replicas to work independently [40]. Both entities can work offline. When they are connected, the merge replication agent checks for changes on both sets of data and modifies each database accordingly. If transaction conflict occurs, it uses a predefined conflict resolution algorithm to achieve consistency. Merge replication is used mostly in wireless environments.

- Statement Based Replication: The statement based replication intercepts every SQL query and sends it to different replicas [58]. Each replica operates independently. To resolve conflicts, Read-Write queries are sent to all servers where as read only queries can be sent to only one server. This enables the read workload to be distributed. Statement based replication is applicable for optimistic approaches where each cache maintains the same replica.

Replication strategies mentioned above need to be implemented using specific techniques. Materialized view is a transactional replication technique which is adopted in different database management systems, such as Oracle database. Database replication in the current model of the Government Centralized Database is achieved using Materialized Views techniques.

2.7.2 Materialized Views

A view can be described as a union of subparts from different data sources. The view should have the same properties as the original data, e.g. when original data is changed

the view is also changed. The purpose of a view is to increase performance on data that is commonly accessed [22].

A materialized view (MV) is similar to a view but the data is actually stored on disk. Materialized views are often used for summary and pre-joined tables, or just to make a snapshot of a table available on a remote system. A MV must be refreshed when the data in the underlying tables is changed [41]. Materialized views can be compared to a cache; they provide fast access to data and since query speed is a critical issue in some applications it is not effective to recomputed the view for every query. Materialized views are frequently used in application such as data warehousing, replication servers and also for query optimization [22].

Since a materialized view is a copy of data, it can contain dirty data if the original data is updated. The progress for updating a materialized view with the correct information is called view maintenance [22].

Incremental view maintenance is a technique used to maintain a view without re-computing it from scratch if changes have occurred in data. It is often cheaper to compute and propagate the changes to the view for the update of the materialization. This, since the size of the base relation and the view, compared to the changes are very small. There have been many proposals of strategies to support incremental view maintenance [22].

2.8. Evaluation Methods

In this section we introduce evaluation methods used to testify software architecture and validate the proposed framework. The evaluation considers quality attributes and is based on a method used for analyzing software architectures against quality attributes and is called Architecture Tradeoff Analysis Method (ATAM). Additionally, the evaluation considers a proof-of-concept method for the validating the realization of the proposed framework.

2.8.1 Software Architecture Evaluation Using ATAM

Here we address software architecture evaluation methods and the ATAM approach as it is the method to be used for the architecture evaluation. Architectural design decisions

determine the ability of the system to meet functional and quality attribute requirements. In the architecture evaluation, the architecture should be analyzed to disclose its strengths and weaknesses, while eliciting any risks [8].

Two comparison criteria for software architecture are identified, namely, early software architecture evaluation and late software architecture evaluation. In the thesis work we are using the former one for evaluation, since we are proposing a framework and it does not have detailed architecture components. Early software architecture evaluation has the following features:

- It is a scenario-based evaluation and do not need data measured from implementation.
- It does not require metric usage.
- It can be based on mathematical model, simulation based or experience based.
- It requires the participation of the stakeholders.
- It has several methods that are discussed in a large volume of software engineering literature, among these are: Software Architecture Analysis Method (SAAM), Architecture Level Maintainability Analysis (ALMA), Architecture Tradeoff Analysis Method (ATAM), and Performance Analysis of Software Architecture (PASA)

ATAM is the most suitable for thesis framework evaluation among the mentioned methods since its superior to SAAM, and both ALMA and PASA evaluate for attributes other those to be evaluated in the proposed framework.

The ATAM method -developed by the Carnegie Mellon Software Engineering Institute (SEI)-, relies on the elicitation of quality attribute scenarios from a diverse group of system stakeholders. The ATAM is an enhanced method for the SAAM. The purpose of the ATAM is to assess the consequences of architectural decisions in light of quality attribute requirements [30].

The ATAM gets its name because it not only reveals how well an architecture satisfies particular quality goals (such as performance or modifiability), but it also provides insight into how those quality goals interact with each other—how they *trade off* against each other. Such design decisions are critical; they have the most far-reaching

consequences and are the most difficult to change after a system has been implemented [30]. The method was created to uncover the risks and tradeoffs reflected in architectural decisions relating to those quality attribute requirements. Quality attributes, also known as nonfunctional requirements, include usability, performance, scalability, and Interoperability, and so on. One of the positive consequences of using the ATAM is a clarification and concretization of quality attribute requirements.

Quality attribute scenarios give precise statements of usage, performance and growth requirements, various types of failures, and various potential threats and modifications [7]. Once the important quality attributes are identified, the architectural decisions relevant to each high-priority scenario can be illuminated and analyzed with respect to their appropriateness [5]. The resulting analysis yields:

- Risks: architectural decisions that might create future problems for some quality attribute. A sample risk:
- Non-risks: architectural decisions that are appropriate in the context of the quality attribute that they affect.
- Tradeoffs: architectural decisions that have an effect on more than one quality attribute.
- Sensitivity points: a property of one or more components, and/or component relationships, critical for achieving a particular quality attribute requirement.

The ATAM analysis of the quality attribute scenarios gives insight into how well a particular SOA-based architecture satisfies the particular quality attribute goals of these scenarios and how certain quality attributes interact with each other in an SOA context. The ATAM focuses on quality attribute requirements. The major goals of ATAM are to [30]:

- Elicit and refine a precise statement of the architecture's driving quality attribute requirements.
- Elicit and refine a precise statement of the architectural design decisions.
- Evaluate the architectural design decisions to determine if they satisfactorily address the quality requirements.

The ATAM method is performed into nine steps which are [8]: 1- Present the ATAM, 2- Present the business drivers, 3- Present the architecture, 4- Identify architectural

approaches, 5- Generate the quality attribute utility tree, 6- Analyze the architectural approaches, 7- Brainstorm and prioritize scenarios, 8- Analyze architectural approaches, and 9- Present results.

These steps are typically carried out in two phases. Phase 1 is architect-centric and concentrates on eliciting and analyzing architectural information. This phase includes a small group of technically oriented stakeholders concentrating on Steps 1 to 6. Phase 2 is stakeholder-centric, elicits points of view from a more diverse group of stakeholders, and verifies the results of the first phase. This phase involves a larger group of stakeholders, builds on the work of the first phase, and focuses on Steps 7 through 9 [28]. The analysis prescribed in the ATAM is not meant to be precise and detailed; it does not provide numerical values for different qualities. The key is to elicit enough architectural information to identify risks, which result from the correlation between the architectural decisions and their effect on quality attributes [8].

2.8.2 Framework Validation Using Proof-of-Concept

To validate and verify the correctness of the framework a proof of concept method can be used. A Proof-of-Concept can refer to a partial solution which involves a scenario or case study of the framework that involves a relatively small number of users acting in business roles to establish whether the system satisfies some aspect of the requirements. A Proof-of-Concept is used to validate that the architecture and ensures that it meets the requirements and identify issues and areas for improvement.

In a Proof-of-Concept, a showcase application is developed demonstrating the correctness of the implementation and the usefulness of the concept, the Proof-of-Concept is used to validate the correctness of the implementation. A proof-of-concept is usually small and may or may not be complete. By contrast, the objective of a proof of technology is to determine the solution to some technical problem, such as how two systems might be integrated or that a certain throughput can be achieved with a given configuration. No business users need be involved in a proof of technology [62]. It is a validation of the applications or concepts formulated. In a proof of concept a partial software solution running code that realizes key usage scenarios, where those scenarios provide proof that the concepts will work as planned.

A Proof-of-Concept provides laboratory measurable and functional proof that validates technical choices. The Proof-of-Concept consists of technical solutions to key technical issues; it is realized by a specific technology mix solution. The Proof-of-Concept evaluation method will be used as an approach to validate the proposed framework by implementing a prototype.

2.9. Technical Foundations Discussion

The technical foundations presented so far can be tight together as depicted in Figure 2.7. It is clear how such theories and techniques can be grouped and associated with each other to lead a SOA solution for the thesis problem. It is apparent from the Figure 2.7 that SOA solutions are linked to different techniques and concepts, such as ESB and Web Services. Web Services in turn are associated with WSDL and SOAP. Since any Web Services relies on WSDL and SOAP concepts. Service Registry which is part of a typical SOA architecture is linked to UDDI, and usually implemented as a web service. The database replication techniques will be used in the web services that implement the replication.

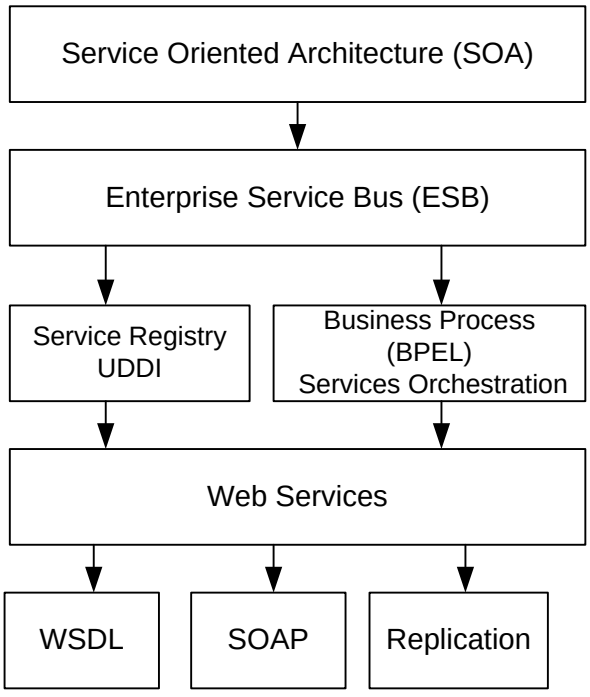

Figure 2.7: Technical Foundations Dependencies

Chapter 3

Related Works

In this chapter different related works are studied and investigated. The related works are introduced and analyzed with respect to the thesis problem to show how far these works address the requirements of our thesis problem. Parts of the related works can be a basis for solving the thesis problem, but no one can present a complete solution. The related works and researches are focused on SOA and how to apply its principles towards the integration of applications and data sources. The related works presented addressed the area of SOA and how it is related to e-Government, data integration, data warehouses, and information systems. The following sections are presented to discuss related work to thesis: SOA research directions, Information as a Service, SOA and Data Integration, SOA and Health Information Systems Integration, and SOA and e-Government Systems.

3.1 SOA Research Directions

In this section we present the research directions in service-oriented computing. One of the main papers that addressed research topics of SOA was by Papazoglou et al. [57], in which he addressed the directions in the research roadmap of SOA into 4 layers, which are: service foundation layer, service composition layer, service management layer, and service engineering layer. While the four layers are related to our research topic, the service foundation is the one that is mostly related and close and should be considered. The state of art for the service foundation layer is to have an infrastructure for Web services and SOA that is realized based on the concept of ESB. One of the challenges in this layer is to have an infrastructure that support for data integration, where the infrastructure has the ability to provide consistent access to all the data by all the applications that require it, in whatever form they need it. This challenge is addressed by this thesis work, since we are seeking to integrate different data sources into a unified one, and provide access support for them.

3.2 Information as a Service (IaaS)

In [16] Dan et al. proposed modeling and realizing information as a service. They discussed the application of SOA principles which are: reusability, flexibility, and interoperability, to enable the use of Information as a Service (IaaS) and to unify information and process service approaches to SOA. They also addressed the benefits in unifying information and process service approaches to SOA, and key research challenges in modeling and realization for IaaS in the context of SOA development. They stated the features of IaaS as loose coupling to data stores and data model, reuse of data access logic, support of data governance, separation of concerns, ease of development of data service by DBA, and optimization of data access Logic. Still their proposal did not present security issues e.g. authorization and authentication, or distributed databases and replication as a service.

3.3 SOA and Data Integration Solutions

The goal of this section is to show different approaches for data integration based on SOA concept. Wang et al. proposed in [72] a model for dynamic data integration based on SOA. The model is dynamic and application oriented, component oriented, and service oriented for data source. It provides service oriented architecture for the integration of heterogeneous data and sharing, and integration of data sources between different platforms is realized. The idea is to integrate data dynamically without building the temporary centralized database. The model has 4 layers, namely uniform data accessor, XML view and SOA mapper, processing engine of XML data integration view and XML data integration view. Figure 3.1 depicts the proposed architecture.

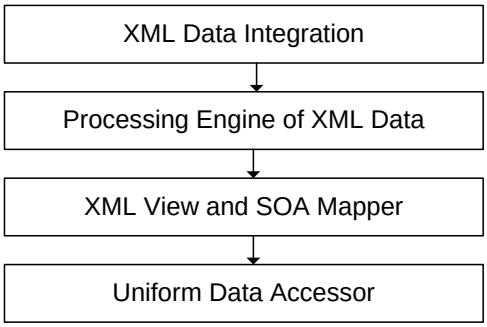

Figure 3.1 : Dynamic Data Integration Model Architecture[72]

Even though the proposed model solves the problem of data integration, still it cannot be applied to solve our thesis problem which is the requirement to have a central database.

In [46] Minguez et al. proposed to reuse and integrate Champagne into a service-oriented architecture in order to benefit from SOA principles. Champagne [64] is data propagation system based on XML technologies and event driven propagation and transformation scripts that ensures the Interoperability of enterprise applications at the data level, but provides a tightly-coupled integration of applications and lacks flexibility.

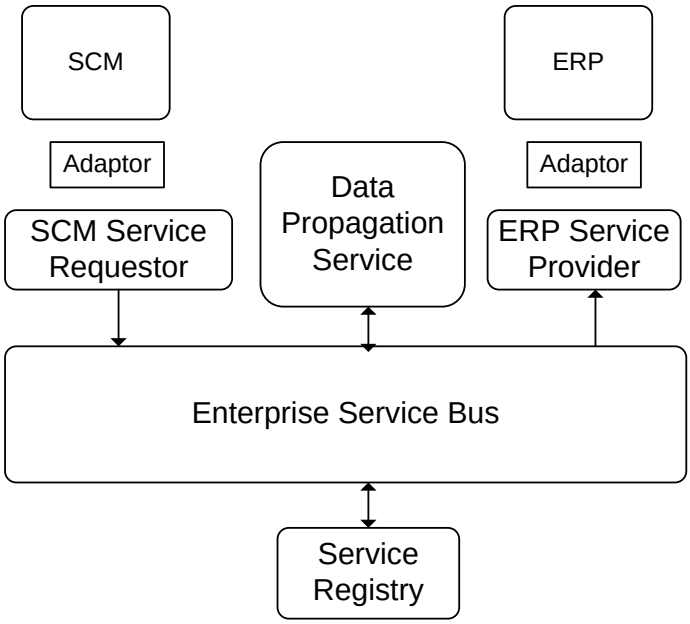

Figure 3.2: Data Propagation System

Figure 3.2 presents the data propagation system. The system is restricted to using champagne software and data access as a service is not presented in the model.

Changyu et al. in [11] proposed a design for intelligent traffic management data center based on SOA. They offered specific design plan and its implementation. Their work was motivated by the inability of the traditional data centers to fully meet modem traffic management work developments. They proposed a framework that includes 4 layers which are: data service layer, application service layer, integration service layer, and external published integrated layer. Their proposal does not address replication and security issue.

3.4 SOA and Health Information Systems Integration

In this section we present different solutions for diverse health information system. Delin Q. in [17] proposed a design of medical insurance supervision system based on active warehouse and SOA. The goal of the system is to improve real-time performance of data analysis and queries, and tries to solve the worldwide problem of the medical expenditure control, by improving the efficiency of medical expenditure supervision. The proposed design would overcome the shortcoming of traditional data warehouse which is not suitable for daily operation decision-making support. The proposal is based on dividing the data warehouse into two divisions static and dynamic. The proposal presents an active data warehouse which has two major functions, active data access and real-time or near real-time data analysis. The methods used for active data access are Extract Transform Load (ETL) and EAI.

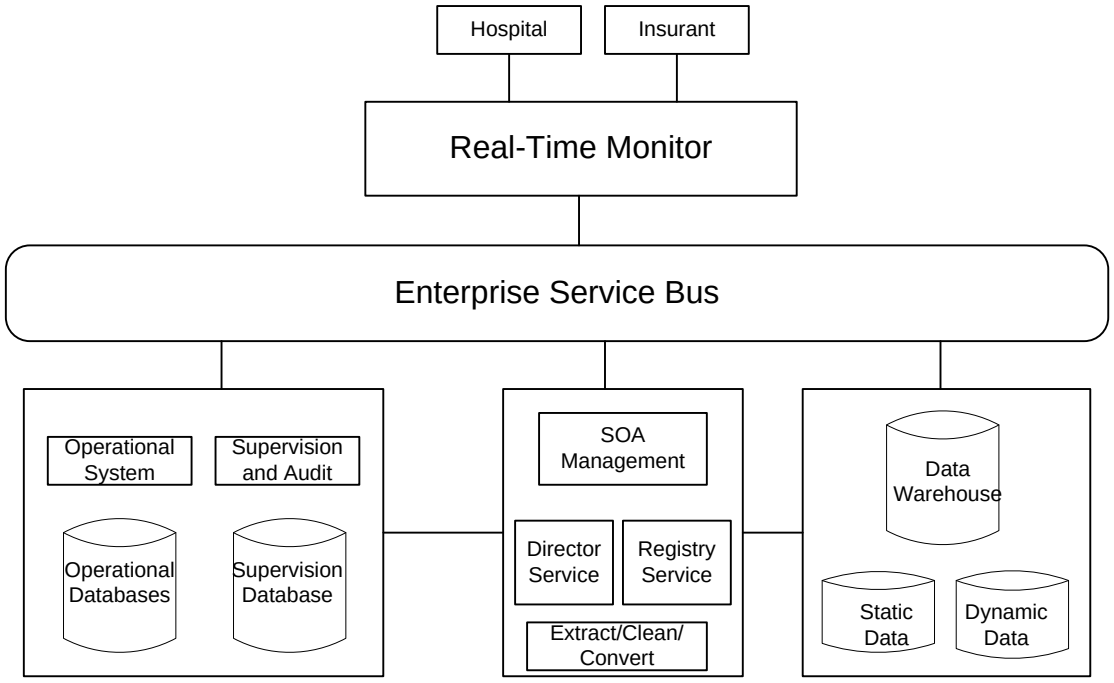

Figure 3.3: Medical Insurance Model [17]

Figure 3.3 depicts the medical insurance model. It's apparent that the model did not address replication between databases as well as security issue, and it is domain specific for insurance database. As well as the data extract is classified into two parts static (periodically extracted) and dynamic (based on triggers in the operational system). This model cannot be applied to solve the thesis problem because replication should be

initiated from the central database and not from the operational database systems in the ministries.

Strähle et al. in [69] proposed a technical concept that employs a SOA as an IT platform and ecosystem to handle different modalities, devices, and data streams in the operation room. They proposed SOA for networked medical devices and integrating legacy medical devices in this network. Their proposed solution addresses accessing medical devices through Web services which are connected to the ESB. They applied the orchestration concept to provide data and modalities to front-end applications. Although their solution provides a SOA based integration for different data accessed from various devices and realized using an ESB, the solution can't be used a solution for our problem. This is because replication and access to different databases issue is not addressed. Figure 3.4 depicts the architecture of their solution.

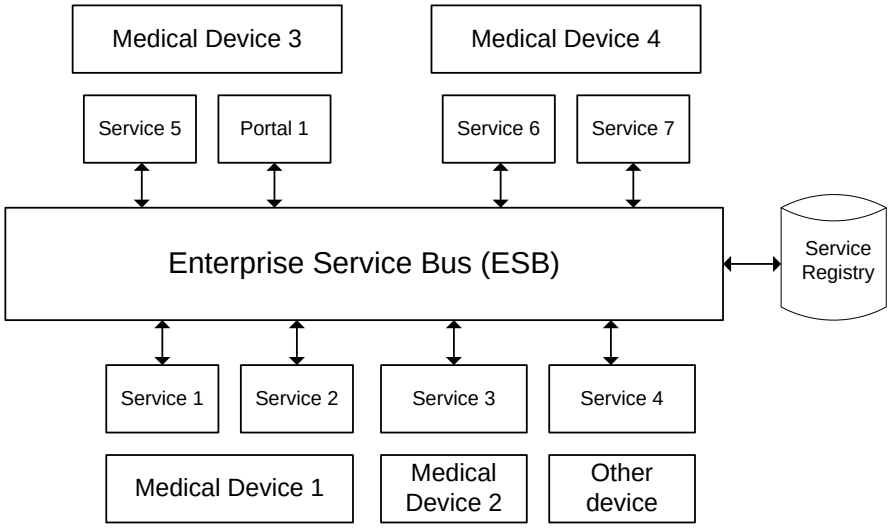

Figure 3.4: SOA Based Network of Medical Devices [69]

3.5 SOA and e-Government Systems

In this section we address different SOA based solution for e-Government systems. Ma proposed in [37] a model for a service-oriented e-Government support platform for the integration of application and data known as (SoGoSP), the proposed model is able to effectively integrate applications and data from various business systems deployed in e-Government external networks and e-Government internal networks. The model consists of four layers, which include service integration layer, service support layer,

common service layer and application layer. The service integration layer supports adapting heterogeneous e-Government resources into standardized services by transforming or wrapping method. The service support layer is the Enterprise service us and is responsible for management of services, such as service routing, message transmission, service monitor, security authentication and service transaction. Common service layer presents business bus which role is to transform service into component used in application layer. In the application layer more than one application system to deploy in the (SoGoSP), such as cooperative office system, administrative examination and approval system, document transmitting system, electronic supervision system, meeting notice system, bills and measures system. The application bus is responsible for the management of applications. Figure 3.5 depicts the system structure of SoGoSP.

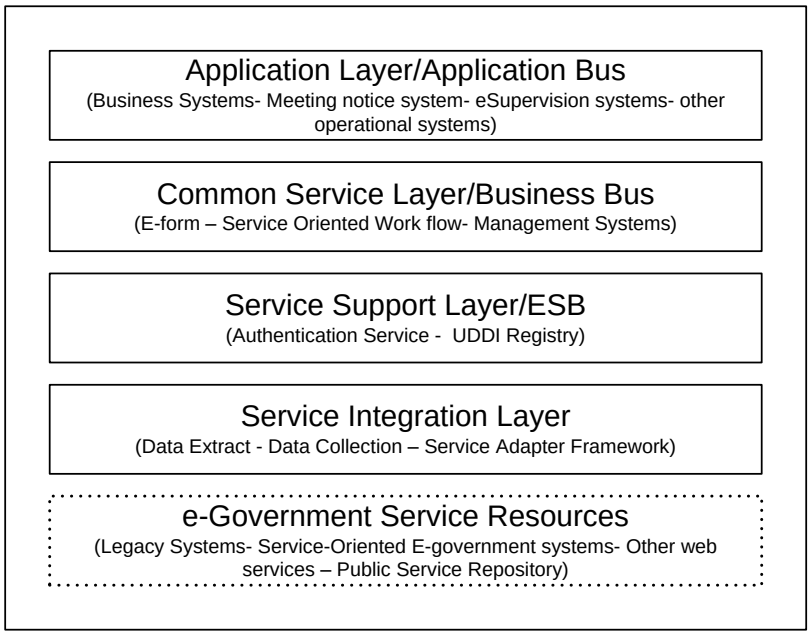

Figure 3.5: System Structure of SoGoSP [37]

The mentioned system does not fulfill the requirements of the thesis work because the replication between databases is not addressed and the scope of the system is the e-Government support as a whole, while our thesis work is focused on data access and integration into a centralized database.

Guo et al. presented in [21] a solution to data integration problem between heterogeneous databases. The solution is based on constructing data center with Web Service technique and XML schema which can give a good solution to problems with business logic method invocation and transparent data exchange in low layer. It is

based on the definition of a unified data definition rules, exchange protocols, and data invocation method management so as to create a distributed data center separated into individual e-Government departments or to create a centralized physical data center. The method is to construct a sharing database with all the needed shared data. The data source should guarantee data consistency and the database using the shared data should guarantee the native data are consistent with that of the shared database.

The users can easily search and subscribe data from each database by a unified interface Figure 3.6 shows the proposed system structure.

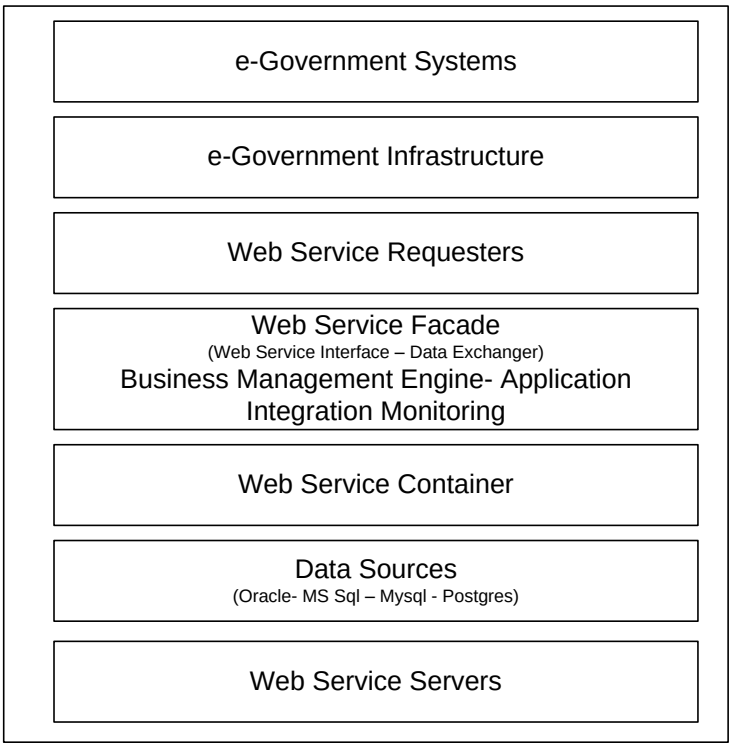

Figure 3.6: Data Center Structure [21]

Even though the structure of the system addresses security and distributed issue for the data, it provides methods for data discovery rather than service discovery with database backend, and it does not provide a solution for database replication, but rather centralizing metadata about the data sources.

Yang et al. proposed in [75] a SOA-Based platform for water resource information exchange named Water Resources Information Services Platform (WRISP). The platform is designed to be a kernel and hub for information exchange between different agencies and to provide a one-stop service website to get the application of eRiver information. Thus the platform would solve the problems of integrated information

from different departments and agencies. It is based on acquiring the instant information of different rivers. The proposed platform is composed of 8 modules which are: Service Entrance of platform, Directory Services module, Authentication and Authorization module, Encryption and Decryption module, Information Interchange Infrastructure module, Service Integrated module, Administration module, and Service Adapter. Figure 3.7 depicts the proposed framework.

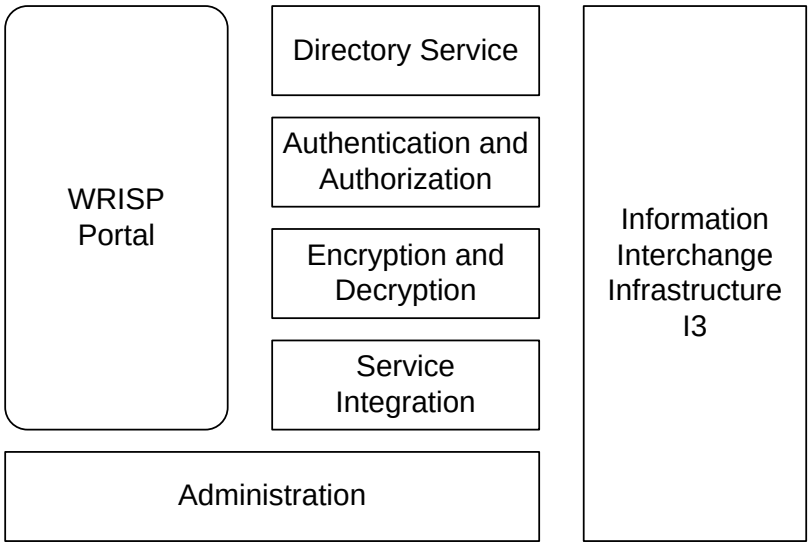

Figure 3.7: WRISP Framework [75]

The proposed framework does not address replication issue, and it considers acquiring of data from different rivers applications.

Yunliang et al. in [76] addressed the requirements of abstracting, sharing and integrating the existing multiple heterogeneous information management systems in the e-Government information technology by introducing an architecture of information management platform based on SOA. Their proposed system is based on service components extracted from Government Resource Planning, and framework technology is based on J2EE, ESB, Ajax front-end access. Their proposal is focused on process application integration and does not address distributed database replication into a central one.

Chapter 4

Evaluating the Current e-Government Integrated Central Database Model

This chapter is dedicated to evaluating the current Palestinian e-Government Central Database model which is one of the core components of the Palestinian e-Government Technical Framework and will also be presented. Data access and replication of the Central Database model is discussed, and the pros and cons of the model are addressed.

4.1 Palestinian e-Government Technical Framework

The current Palestinian e-Government Technical Framework that is adopted by the Ministry of Telecom and IT – Gaza [47] has four layers with different components that need to be realized in order to have a fully functional technical framework for the Palestinian e-Government.

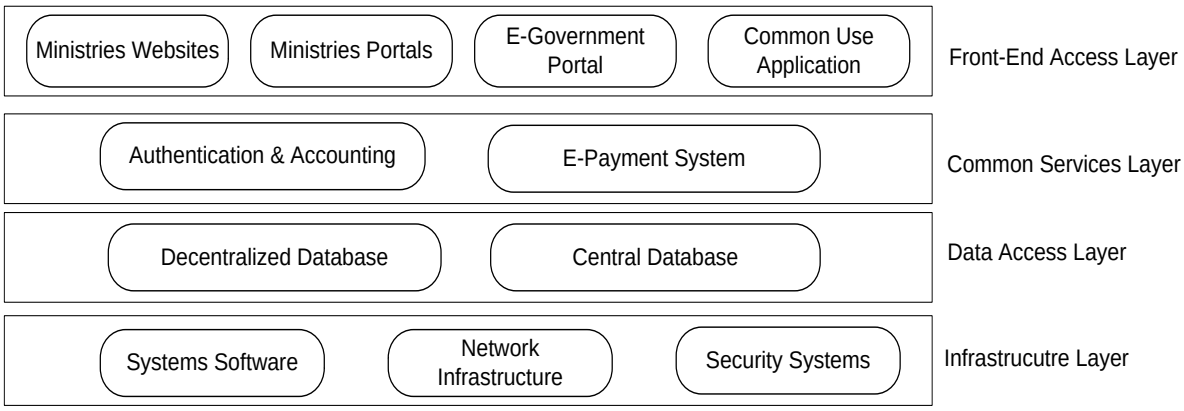

Figure 4.1: Palestinian e-Government Technical Framework

Figure 4.1 depicts the e-Government Technical Framework and its main layers. The Central Database is one of the components that lie in the Data Access layer of the e-Government Framework. The importance of having such component emerges from the fact that e-Government processes and services heavily involve and require exchange of data between different governmental institutes.

4 Evaluating the Current e-Government Integrated Central Database Model

The e-Government Framework is composed of four layers: the front end user interface layer, the common service layer, data access layer, and the infrastructure layer.

1. Front-End Access Layer:

This layer presents the access interface that end user interacts with the e-Government through it. It is considered the visible part of the e-Government, and all access to e-Government services can be achieved via interacting with this layer. This layer includes application such as e-Government portal, ministries websites as well as ministries applications.

2. Common Services layer:

This layer provides front end layer service providers with services that commonly needed by e-Services, such as authentication and e-Payment services. Authentication service authenticate citizens requires access to authenticated services against a single authentication repository, such service is not incorporated in the front end systems, but rather developed as common service to be used by front end systems. Also this layer provides e-Payment capabilities to ministries portals, which is needed since some services incur charging on the beneficiary of it.

3. Data Access layer:

This layer addresses database access gateway, either centralized or decentralized. Front end services rely heavily on this layer. For example, online job submission service, requires access to data from different sources, which turn to be in the Central Database. We further elaborate on this layer in section 4.2.

4. Infrastructure layer:

This layer includes physical and low level software components, such as government private network, operating system and services, and security systems. These components present the interface with networking devices and functionalities such as hosting and collaboration services, firewalls and intrusion detection and prevention systems. The private governmental network is a core element in this layer because of its interconnection capabilities for government offices locations. This is because part of the inter-ministries traffic should be carried over private, independent, and secure link. The software parts of this layer includes the operating systems and their low level services

such as Web hosting and email provision capabilities, database hosting, systems and network management.

The overall e-Government Technical Framework is not fully implemented and parts of it still need to be realized [47]. Also the interaction between layers is not well defined in terms of access protocols or standards. SOA can be introduced here to provide standard way of Interoperability between e-Government components, both software and hardware based components, though discussing how to change the current e-Government Technical Framework into SOA is not in the scope of the thesis. In this research we are proposing to use bottom-up approach in adopting the SOA solution in the e-Government initiative, in which we start from the components rather than from the overall framework itself.

4.2 Analysis of the Current Integrated Central Database Model

The database access layer in the e-Government Technical Framework, discussed in previous section, comprises both centralized database and decentralized databases. In decentralized databases every ministry uses its own database for its own applications and business processes and it has full control over the database. For a ministry to have flexible and efficient information systems there is a need to access data from other partner ministries. The Central Database comes as a solution to sharing and integrating data between various ministries. The focus in this research is on the Central Database in the Data Access layer (Figure 4.1), which is an important part of e-Government Technical Framework; this is because it has a vital role in building and integrating e-Government services. Figure 4.2 depicts the current implemented and used Integrated Central Database model which is approved by the Palestinian Government Data Integration Committee, which is a governmental committee that oversees and recommends the regulations and guidelines for the Integrated Central Database [52].

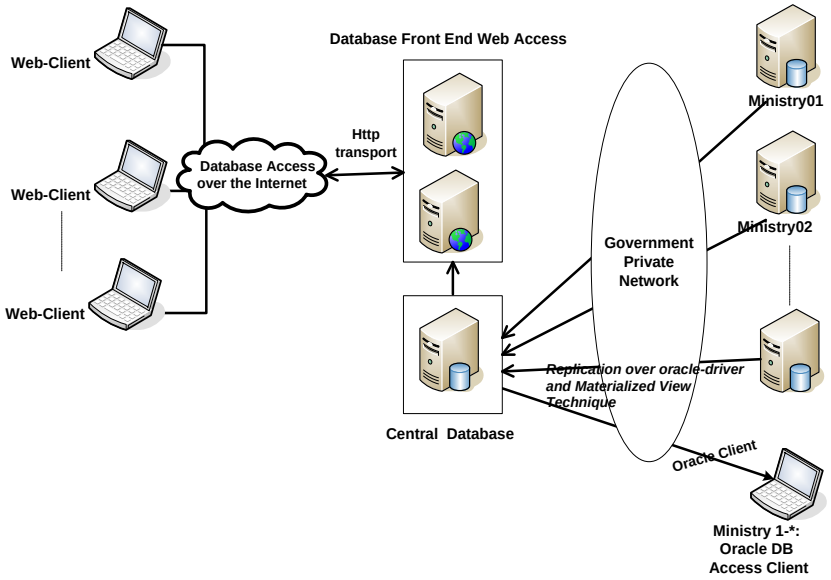

Figure 4.2: Current Integrated Central Database Model

The Central Database can be thought of as a broker that integrates data from different sources and then allows the exchange of data between partner ministries through it. The current model for the Central Database relies on database replication and synchronization techniques as the low level infrastructure to maintain the Central Database and to keep its content up to date.

The main characteristics of this model can be classified into three categories which are: access, replication, and management and monitoring. Following the characteristics are explained.

- Database Access :
  - The Central Database access mode is read-only for ministries use, since it has a one-way replica for ministries databases, the writing to the database is done at each ministry that owns the data.
  - Web access to the database is done through Web applications connecting to Central Database over Oracle connectivity driver.
  - Direct access to Central Database is allowed from the governmental private network, and requires Oracle connectivity driver.
  - No direct access to the Central Database from the Internet which is classified as un-trusted zone.
  - Client access to the Central Database is realized using Oracle stored procedures, with predefined parameters.

– Only synchronous mode of invocation is available.

- Database Replication:
  – The Central Database operator has read-only access to each ministry's database that is replicated to the Central Database.
  – Replication between ministries and Central Database is achieved using Oracle utilities and tools such as materialized views and database links.
- Management and Monitoring:
  – Monitoring the Central Database is performed based on database parameters, and the monitoring system is not a proactive one.
  – Governance issues are limited to managing the main functionalities of the database access. The current deployment of the Central Database lacks governance features such as enforcing Quality of Service (QoS), usage obligations, Service Level Agreement (SLA), access metric, responsiveness criteria.
  – Security policies are implemented at both network and database access level.

The above mentioned characteristics do not impose clear constraint on using SOA solution in the Central Database. There are no old fashioned legacy applications or rigid connectivity access mode to the used database, which make SOA a suitable framework for realizing the Central Database model.

The current Central Database model is being criticized for the various limitations some of them are listed below:

- Replication between the Central Database and the ministries databases can only be achieved between Oracle type databases using the Materialized Views technique. Hence synchronization and consistency is restricted to the options provided by the Materialized Views. This imposes inflexibility on database usage and force ministries to using proprietary commercial database systems.
- Access to the Central Database is restricted to Oracle connectivity driver, which decreases the level of interoperability.
- Direct access to database procedures is achieved only with government private network and through Oracle standard sql port, which undermines the flexibility and accessibility.

- Central Database has a read-only access mode which undermines the capabilities of the database.
- There is no standard way for the describing, finding and invoking the procedures defined in the Central Database which are accessed by its clients, this leads to less flexibility and more management overhead.
- System monitoring, management and security assurance is built on Oracle database itself.
- Limited governance issues are addressed in the system.

The above mentioned limitations undermine the flexibility, interoperability, scalability, manageability, and governance of the Central Database, such features can be achieved if we adopt SOA solution. However, we only consider building a SOA-based solution that accomplishes the three quality attributes: *interoperability*, *flexibility*, and *manageability* (see section 1.5).

Chapter 5

SOA-based Framework

In this chapter we propose an Integrated Central Database model based on SOA concept. As discussed in section 2.1, the reason behind proposing to adopt SOA is because of its open architecture and platform standards that cope with heterogeneous systems and applications in order to achieve high degree of scalability and flexibility.

Web services will be used as the main building blocks to realize SOA architecture. The fast adoption of Web services emerged from the maturity of XML-based Web services standards such as SOAP and WSDL [2]. To realize the concept of SOA and to achieve a suitable and manageable integration infrastructure for Web services, the hub and spoke approach using ESB will be used [55]. The ESB is the middleware that integrates the components of the SOA concept; it integrates the applications, services, and the registry [31],[53]. In reference to the discussion presented in section 2.6, the ESB [15] will provide functions such as: Routing, Message transformation, Protocol transformation, Service mapping, Service choreographing, Service orchestration, Transaction Management, and Security. In next section we present the SOA-based Integrated Central Database requirements, the proposed framework architecture, components interactions, and ethical issues related to the framework.

5.1 SOA-based Integrated Central Database Requirements

The requirements for the Central Database need to be defined ahead of presenting the proposed model. To overcome the shortcomings of the current model as mentioned in section 4.2, the requirements are specified as follows:

- Accessibility Mode:

The Central Database accessibility should be based on standard connectivity rather than proprietary commercial software access mode, such standards are XML, SOAP and WSDL. In this case, services would access the underlying database without using its access driver, and hence services would be specific database type independent.

- Replication:

Government ministries need to replicate and synchronize heterogeneous database type such as: Oracle, MySQL, MS-SQL, MS-Access, with the Central Database. Hence different replication options should be provided for the diverse types of used databases, and only the replication service to be aware of such diversity.

- Governance:

The Central Database should operate around the hour since individual ministries IT infrastructure lacks the ability to do so. The Central Database should be governed through QoS and Service Level Agreement (SLA) since different ministries rely on it providing eServices which should operate without interruption as well as provides a level of responsiveness.

- Management and Monitoring:

Monitoring and management should be separated from the application logic and database procedures access. Logging and performance metric recording should be implemented.

- Security:

Security must be assured and should be managed centrally and imposed on all database access through the different services. Security policies to be defined and enforced and must not be configured at the underlying database level only, but also at the service level.

- Reachability:

Access to government Integrated Central Database should be allowed to both government and non-government institutes. The access should be allowed through the government private network, as well as the Internet.

5.2 SOA-based Integrated Central Database Architecture

To realize the requirements discussed in section 5.1 for the proposed SOA-based framework of the Integrated Central Database, different components are presented that constitute the proposed framework. Each component satisfies one or more requirements and leads to the achievement of the goals of the framework. The framework which is SOA based and realized using Web Services and ESB is depicted in Figure 5.1.

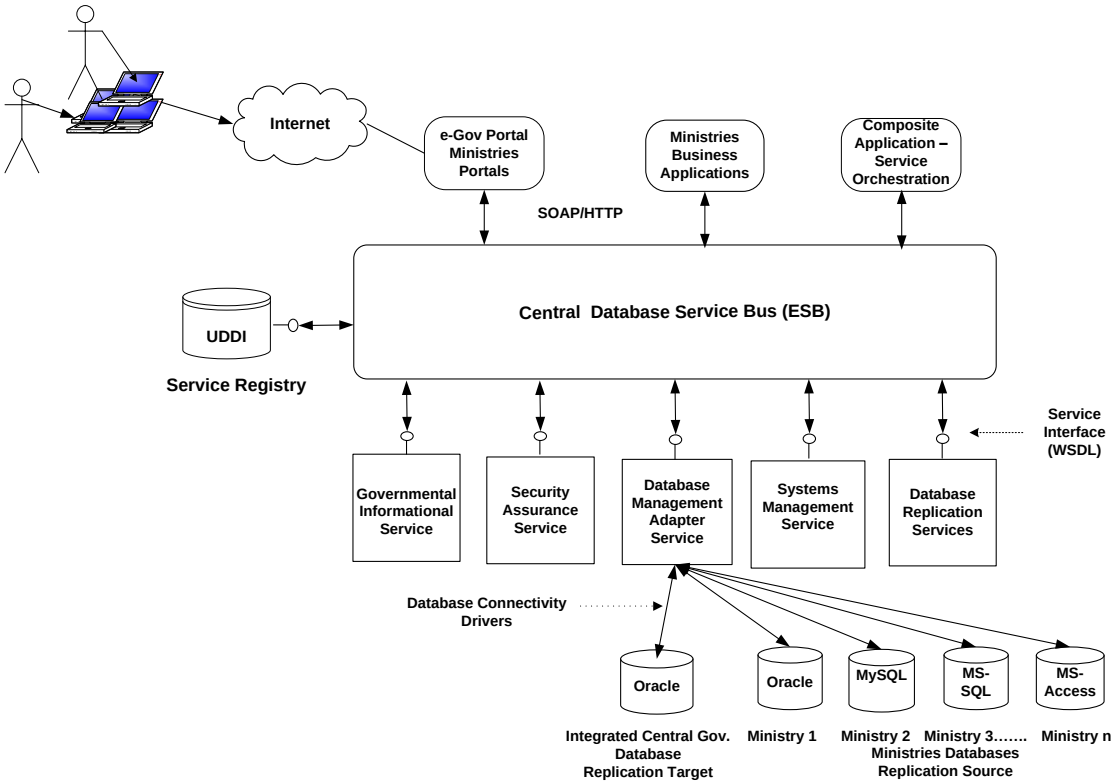

Figure 5.1: Proposed SOA-based Integrated Central Database Framework

The main components and their description are listed and discussed below.

1. Central Database Service Bus:

This bus is considered the central platform of integration between different Web services, and provides routing and transportation features for Web service requests, as well QoS feature for the framework. It will be used and accessed by government institutes via government private network, as well as over the Internet for non-government institutes, hence accomplish the reach-ability requirement of the framework.

2. Service Registry:

The Service Registry will be used to provide a search point of access to services and database definitions and metadata for all services provided by Central Database model. This registry will be based on Universal Description Discovery and Integration (UDDI).

3. Government Informational Services:

These Web services provide access to basic informational queries; these services allow consumers to benefit from government Central Database along with its presentation logic, this will relief them from invoking services that interacts directly with the Central Database and return record sets that need to be manipulated by the developer. For example, Web service that returns social information of a citizen, or employee administrative record.

4. Service Orchestration:

This component is an important part of the Central Database Bus, which is responsible for managing composite services. The composite service is invoked by client and in turn it invokes and orchestrates different services to achieve the requirement of the composite service.

5. Database Management Adapter:

This adapter will allow the Central Database Service Bus, to accept requests for data source from client systems then invoke the relevant adapter to retrieve the data, and return it in a standard format to the requester. It is used to hide the database management details from the rest of the Web services, and it is the only service that communicates directly with the underlying data source and should have database specific connectivity capabilities. This component will accomplish the accessibility requirement.

6. Database Replication Services:

Such services will be used to manage replication between the Central Database and ministries databases, connections types, mode of replications, access permission, etc. are addressed by this service. These services are responsible for achieving the replication requirement.

7. Systems Management Service:

The management service will be used to manage and monitor the Central Database service bus, and Web services. It will collect metrics, provides framework performance reporting capabilities. Both governance and management requirements of the framework are achieved in this service.

8. Security Assurance Service:

This service will insure that security policies are adhered to and will achieve the security requirement in the framework. It will be invoked by different services to add security layer to their functionality. Security functionalities provided are: authentication, authorization, and non-repudiation. This service would carry out the security requirement of the framework.

5.3 Framework Interaction

The interaction between the components is done through the Central Database Service Bus, which will integrate the components and will act as the glue that tight them together, it will route, transport, and format the requests and response of the services, it will also provide service discovery through the registry. The usage scenario in section 7.3 further illustrates the component interaction.

This framework achieves its goals which are Interoperability, Flexibility, and Manageability. The Interoperability, which allows using diverse types of databases, is achieved by having different database types as part of this system and can be part of the replication as well as resource for different governmental information services. The Flexibility, which allows different ways for performing a specific task, is achieved by accessing the information services over HTTP transport which generally uses the port 80 which is normally not filtered by internet firewalls, and hence the access to the Central Database can be both from internal government private network as well as over the Internet. The Manageability, which provides the ability to control and adjust the behavior of the system in response to various circumstances, is accomplished by having metric, performance, QoS as part of the logic in the management services. More about the evaluation of the framework is presented in Chapter 7.

5.4 Ethical Considerations

Ethical considerations should address all aspect of the thesis work, from beginning to end. This is because the e-Government central database usually contains personal and private data, such as payroll data, health records, population registry information, work information, and so on. This would impose to take all measures to achieve the confidentiality of the data. Even though the Security Assurance Service discussed in the previous section provides security for those accessing the informational services, still the following questions are raised:

- What if an authenticated staff tries to access data sources for non-work purposes?
- What if the database administrator goes through database records and uses it for personal profit?
- What if security measures and mechanisms were uncovered disclosed and abused by a legitimate and trusted network administrator?

Most of the time it is very difficult to address the above mentioned problems via technical ways only, and hence comes the requirement to adopt an ethical code of conduct for those who will be interacting the central database. Such ethical guidelines are proposed and adopted by the staff responsible for the management of the Integrated Central Database.

Chapter 6

Framework Prototype

In this chapter the implemented framework prototype is presented, first the prototype architecture, and then the Web Services views. In addition detailed information about the prototype is included in the appendices. The working environment of the prototype is included in Appendix A "Prototype Working Environment", and explanatory technical documentation of the Informational, Security and Replication services, as well the composite applications using BPEL are presented in the appendices B, C, D and E, and the front-end interface is more elaborated on in Appendix F.

6.1 Prototype Architecture

To provide a proof of concept of the framework, a prototype should be implemented. Such prototype would provide a specific usage scenario for the framework, and through such prototype the framework is validated to perform its requirements. Figure 6.1 depicts the top-level run time view of the prototype architecture which was realized as a proof of concept for the proposed framework. From the end user perspective, a user is accessing the front end web interface which provides an access to the services of the Integrated Central Database (*GovDB*). The web interface runs in the context of a web application which interacts with the framework over Web Services interfaces. The functions available in the front end interface are access to the Informational Services, that's query the governmental database for citizen record, and database replication service trigging.

The framework runs in the context of the Java Business Integration (JBI) which the realization of the ESB. The JBI ESB used is OpenESB which is an open source product [50], and selected since the governmental ICT strategy supports the usage of open source software [47]. The JBI has the BPEL Service Engine, Database Binding Component, and Composite Applications for Replication and Informational Services. The Informational Services implemented are four: the Security Assurance Service

(*SecuritySvc*), Citizen Data Service (*CtzDataSvc*), Employee Data Service (*EmpDataSvc*), and Health Insurance Data Service (*InsuranceDataSvc*).

The prototype is composed of three databases: The governmental database type is (Oracle 10g), it will be referenced as *GovDb*, The employee records database with type Oracle 10g, and the data source is from the general personal office, it will be referenced as *EmpDb*, and The Health Insurance data, its type is MySQL, and the data source is from the health insurance system, and it will be referenced as *InsDb*. Access to the database is done over the JDBC. The interaction between Web Services consumer and providers is done over SOAP/HTTP transport. Direct access to the *GovDb* will be using stored procedures which provide more security where database access is not allowed directly and tables and views are not exposed to clients, only predefined procedures with specific input/output are allowed in accessing the *GovDb*.

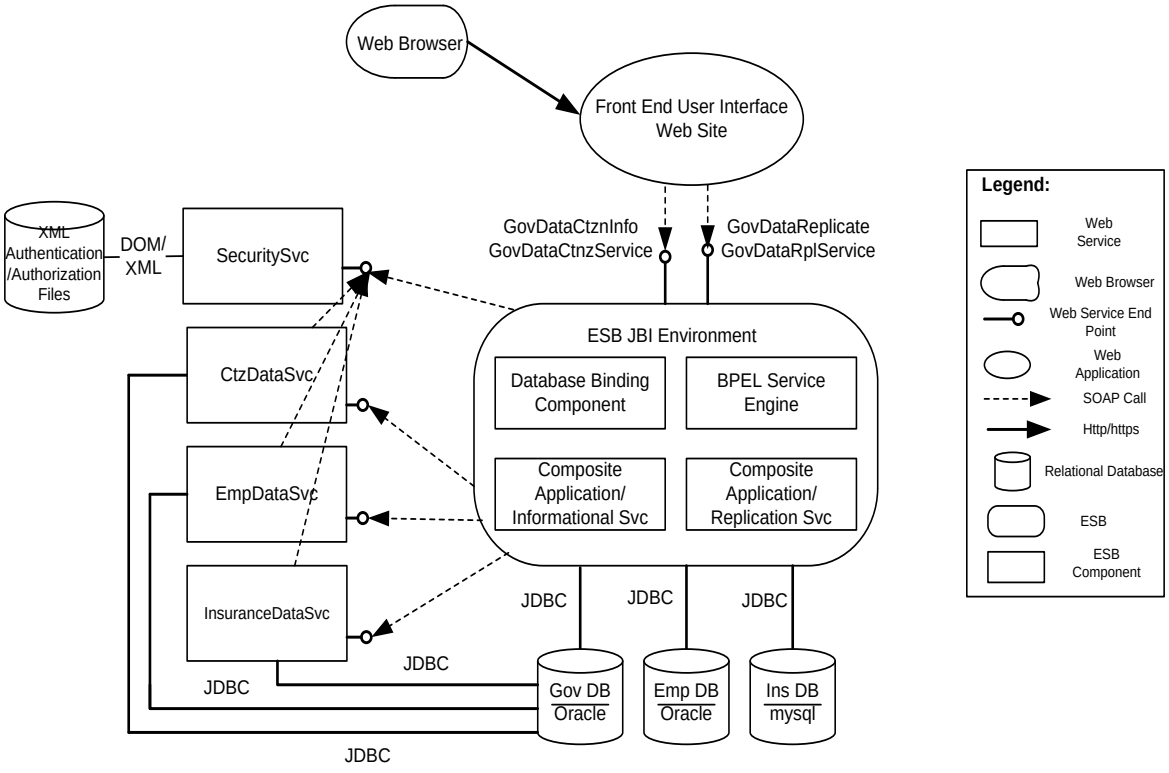

Figure 6.1: Prototype Architecture

The prototype architecture can be associated with the SOA-based Framework proposed and explained in section 5.2. The framework as seen in Figure 5.1 is composed of different components that can be mapped to the parts of the prototype. The importance of the association between the prototype and the framework is that it verifies that the

prototype implementation scope includes the main components of the framework. The parts of the prototype and their counter mapping in the framework are as follows:

- The ESB JBI Environment is mapped to the ESB.
- The Composite Application/Informational Service is mapped to the Service Orchestration.
- The Front-End user interface is mapped to the e-Gov. Portal.
- The services (*CtzDataSvc)*, (*EmpDataSvc*), and (*InsuranceDataSvc*) are mapped to the Informational Services.
- The (*SecuritySvc*) is mapped to the Security Assurance Service.
- The Composite Application/Replication Service is mapped to the Replication Service.

6.2 Web Services Logical Views

In this section we present the logical views of the prototype Web Services. The main functional requirements of the Integrated Central Database are: access to database and replication. Hence we choose to implement the Informational Services and Replication Services. The services implementation is based on top down approach, since we do not have an application or existing systems that performs their functions. We start from the schema XSD, followed by the WSDL, and then the implementation of the service. Implementing of the services would provide a valuable validation environment to prove the correctness of the framework

6.2.1 The Informational Services

The prototype implementation has three informational Web Services, and a composite application which orchestrates two informational services. The informational services as mentioned earlier in this chapter are:

- Citizen Data Service (*CtzDataSvc*): Provides Citizen Population Registry Information for a specific Identification Number of a citizen, e.g. (ID, Full Name, Birth Date, etc.)
- Employee Data Service (*EmpDataSvc*): Provides Employee Record for a specific Identification Number of a citizen e.g. (Employmnet- Job Tile- Salary, etc.).

- Health Insurance Data Service (I*nsuranceDataSvc*): Provide Health Insurance Record for a specific Identification Number of a citizen e.g. (Insurance Type, Expiry Date, etc.).

Figure 6.2 presents the logical view of the Informational Service. Two of the services were orchestrated through a BPEL; the services are the *CtzDataSvc* and *InsuranceDataSvc*. In such case the Web Service client needs only one service access to the composite application to get information from both services.

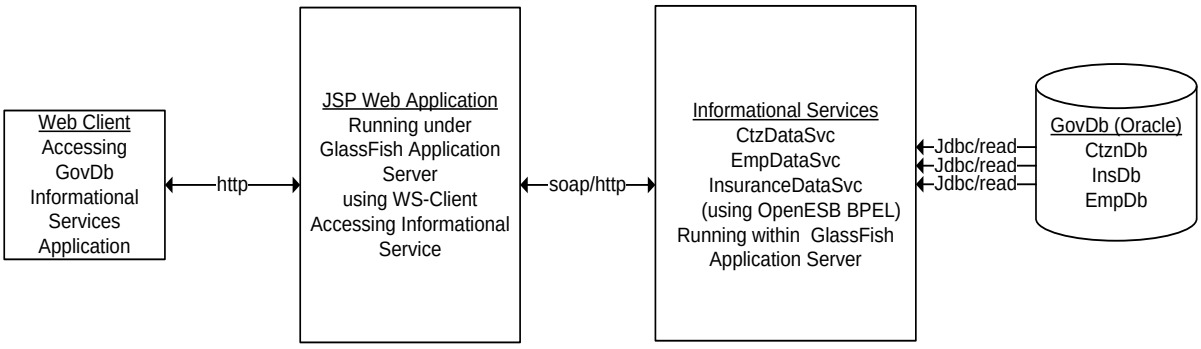

Figure 6.2: Composite Informational Service Logical View

The interaction between the Web Application accessing the Informational Services or the BPEL Composite Application is performed using SOAP over HTTP. For more information about the implementation of the Informational Services such as XSD, WSDL, and BPEL are detailed in Appendix B, and C. The front end access to the Informational Services is done using a web browser, where it interacts with the JSP Web Application that acts as a service requester that communicate with the Informational Services or the Composite Application BPEL over SOAP/HTTP transport. Snapshot and details of the front end access to the Informational services is presented in Appendix F.

6.2.2 The Replication Service

The replication service will use both BPEL and database binding components, and snapshot replication will be used in replication service prototype, in which the whole table is replicated to the *GovDb*. Figure 6.3 shows the logical view of the replication service in which *InsDB* and *EmpDB* are replicated to *GovD*B, access to database is done using the JDBC.

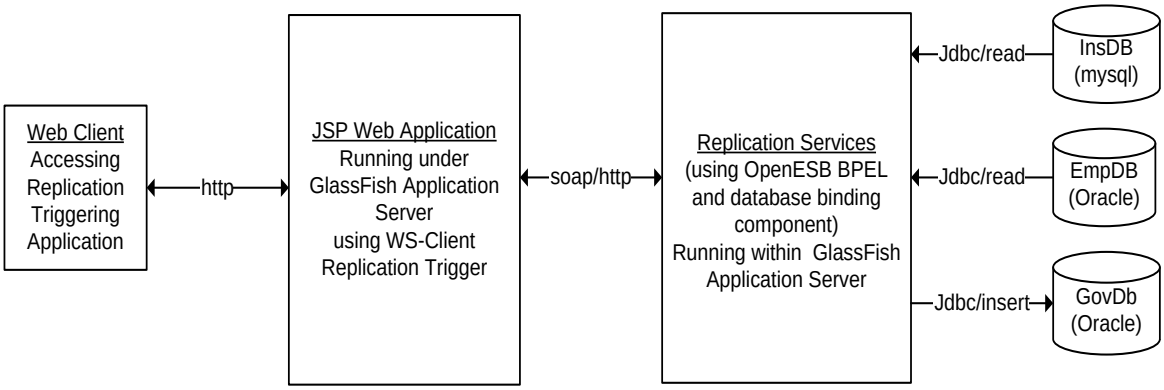

Figure 6.3: The Replication Service Logical View

The replication is triggered from the front end web interface and can be scheduled to be performed based on a schedule. Each of the source databases has a Web Service to read the records in the replication tables or views. Such Web Services uses the database binding component that comes with the OpenESB. There is no need to write any code to read the records in the source tables, just to create a WSDL file that represent the Web Service to access a specific database and integrate such Web Service with the BPEL file that performs the replication. On the target side, were the need to write the records to the *GovDb*, the database binding component is used also to insert the records to the database. Appendix E presents WSDL files for source and target replication database partners, and the BPEL design.

6.2.3 The Security Assurance Service

The Security Assurance Service is used for authentication, authorization and logging of the Informational Services Access. Service authentication will be based on identity management using username and password, and authorization is based on username and IP addressing. Service consumer will be identified by username and password pairs. Access to services will be allowed if the user is allowed to access the service from a specific IP address. This mechanism will ensure that users will access services from allowed IP addresses. This means gaining access to others username and password credentials will not allow for accessing services.

The Security Assurance Service logical view is depicted in Figure 6.4. The service is invoked over SOAP/HTTP transport by the Informational Services, and the service performs both access and accounting functionality. The authorization, authentication and logging information are stored in an XML files and accessed using DOM interface.

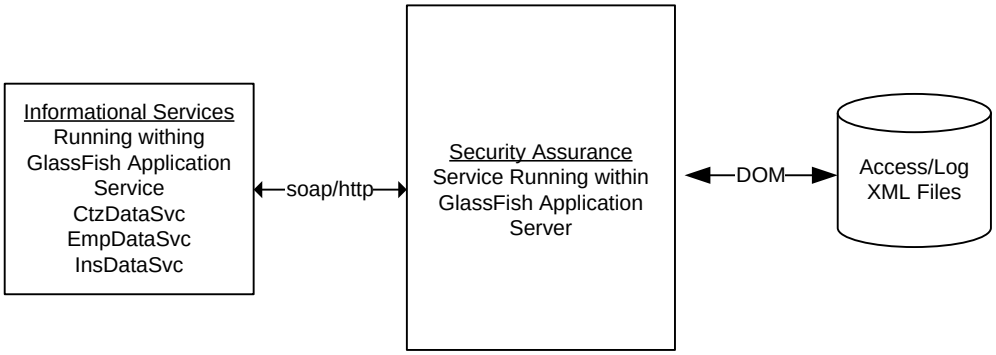

Figure 6.4: Security Assurance Service Logical View

In Appendix D, detailed information about the Authentication, Authorization, and Logging files are presented, and also XSD, WSDL, and snapshot of the Java Code for the Security Assurance are presented.

The flow chart and process flow of the Security Assurance Service is depicted in Figure 6.5.

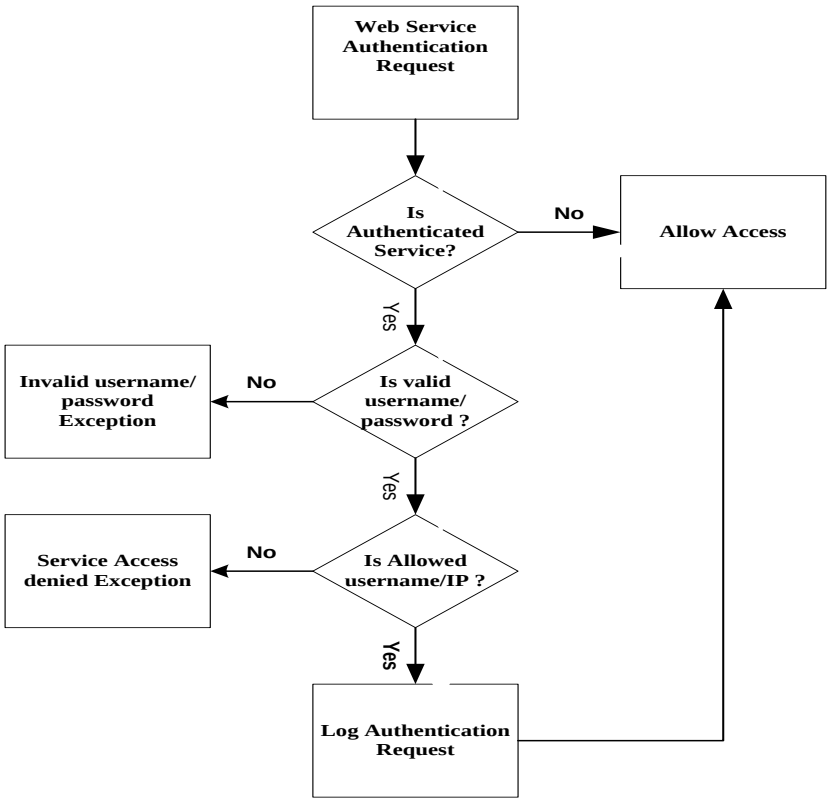

Figure 6.5: Security Assurance Service Flow Diagram

Figure 6.5 shows the Authentication, Authorization, and Logging process flow which relies on username/password authentication and IP access control. This service will use JAX-WS annotation. The business logic of the security service starts by checking if the

service access should be authorized or public. In case on unauthenticated service, the request is allowed. In case of a service that requires authorization, a check is done for authentication of the user, followed by checking the username/IP authorization, and if both succeed then authentication is allowed, otherwise fault is raised.

The security issue is one the items discussed in section 1.5 “Scope and Limitations of the Research” that was determined to be addressed both in the framework and the implementation of the prototype. The implemented Security Assurance Service shows the conformance with the thesis work plan.

6.3 Framework Prototype Summary

Coming to the summary of this chapter, it can be said that the prototype architecture and its implementation as discussed in sections 6.1 and 6.2 provide evidence on how far the prototype fulfills the goals of the framework. It is clearly seen from the discussions presented in the last two sections that the prototype performs the main functionalities of the proposed framework which are: access to the database and replication between diverse database types using Web Services. It was seen that main components in the framework are included in the prototype, and the components of the prototype are mapped to their counter parts in the framework. The use of the ESB allows for the use of the management and monitoring capability that comes with the ESB which can be used to stop/start and monitor the services. The implementation of the security service further allows for using the prototype in a real situation and used by the governmental institutes. The prototype acts as a proof of concept for the validation of the framework, since it showed that the framework can be realized and implemented and hence the concept of the framework is validated.

Chapter 7

Framework Evaluation

The goal of this chapter is to present the evaluation of the framework and its prototype. The evaluation will be conducted against the targeted quality attributes which are Interoperability, Manageability, and Flexibility. The chapter discusses the quality attributes and presents the evaluation of the framework based on ATAM (see section 2.8.1) and validation of the framework concept using a prototype usage scenario.

7.1 Framework Quality Attributes

Quality attributes, also known as nonfunctional requirements, are defined as quoted from IEEE Standard 1061 "Software quality is the degree to which software possesses a desired combination of attributes".

At the Software Engineering Institute (SEI), they believe that the suitability of architecture is determined by the quality attribute requirements that are important to the stakeholders of a system. And hence a given software architecture is suitable for its intended purpose in case of fulfilling the quality attributes. Quality attribute scenarios are usually used to specify quality attribute requirements. Also many quality concerns are primarily handled or strongly affected by the runtime environment. In the thesis work, the framework quality attributes to be evaluated need to be clearly defined, as follows:

- Interoperability:

Interoperability is the ability of software and hardware on various machines from various vendors to communicate with each other without significant changes to either one [34]. It is in greatly determined by compatibility issues between the two platforms involved [8]. For SOA concepts, the following aspects of interoperability have been distinguished [13]: businesses, processes, services, and data interoperability. Businesses and processes interoperability are considered mainly at the organizational level, whereas services and data interoperability require focus on Information Technology issues.

In SOA systems, service consumers and service providers are usually placed in different ownership domains. They are also developed independently on various platforms and loosely-coupled by network. Services are consumed generally without direct management by service consumers [35].

- Manageability:

From the Web Services context, Manageability is defined as a set of capabilities for discovering the existence, availability, health, performance, and usage, as well as the control and configuration of a Web Service within the Web Services architecture, it provides methods for monitoring and managing services and business processes.

It is also defined as an ability which keeps a Web Service and its resources being manageable. The Web Service resource includes the software and hardware components used by the Web Service and a platform on which the Web Service operates. The Manageability capability helps targeting a Web Service, provides a function to monitor operational status, and controls operations along with Web Service protocol. As the Manageability capability enables a service consumer to use Web Services with reliability and stability, it may be an important criterion for one to select a web Service. The Manageability is classified into 3 sub-factors: informability, observability and controllability [35].

- Flexibility:

It is defined as the ease of making changes required by changes in the operating environment, characteristics that allow the incorporation of changes in a design. It is the ability of a design to be adapted to provide functional related capabilities [4]. Also defined as the ease with which a system can be modified for use in applications or environments other than those for which it was specifically designed [24].

To perform the evaluation we need an evaluation method that testifies the framework, and since we are presenting architecture, one of the software architecture evaluation methods can be used. The evaluation of the framework will be based on the Architecture Tradeoff Analysis Method (ATAM), which is an early evaluation approach for software architecture that is scenario-based. The use of a software architecture method for the framework evaluation is justified by the deliverable of the proposed solution which is a SOA based framework and not fully implemented system.

As for the prototype which is a proof of concept for the correctness of the framework a specific usage scenario will be discussed to show the quality attributes achievements

7.2 Framework Evaluation based on ATAM

The quality attributes we are testing the proposed framework for are: *interoperability*, *manageability*, and *flexibility*. The methodology for evaluation the framework will be based on ATAM method (see section 2.8), and we need to set a scenario for these goals and present specific measures for them.

The requirements to conduct the evaluation are evaluation team and stakeholder staff. The Evaluation team typically probes the architectural approaches used to address the important quality attribute requirements specified in the scenarios. The goal is to assess whether these quality attribute requirements can be met. In our case the evaluation team is the administrators of the current Integrated Central Database model and those use it in their applications and systems within the governmental institutes.

Figure 7.1 depicts the overall process performed for performing the framework evaluation based on ATAM method. The evaluation process relies on the evaluation teams, business drivers and constraints, the framework quality attributes, and framework architecture approach. The results of the evaluation are the scenarios and how far the quality attributes are fulfilled. If the scenario presents a non-risk for the quality attributes, the framework is considered achieving the quality attribute.

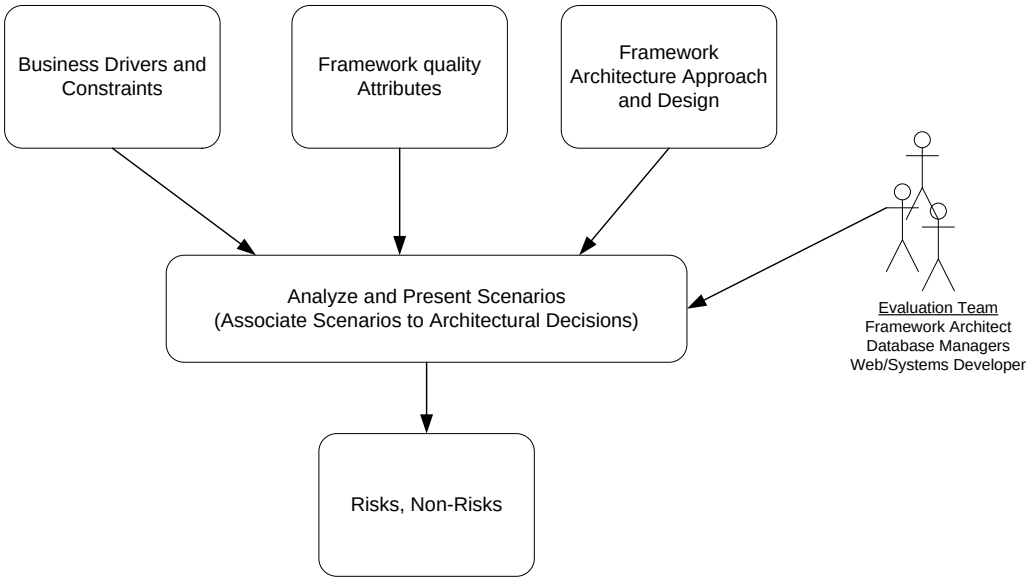

Figure 7.1: Framework Evaluation Based on ATAM

In the following sections we enumerate a collection of quality attributes general scenarios for the three important attributes for the framework which are *interoperability*, *manageability*, and *flexibility*. The main features that support each of the quality attributes are presented which are inducted from the scenarios. The scenarios which are based on ATAM methodology are included in the Appendix G. Each scenario has a question that concerns the quality attribute and has the prompt that shows the framework answers for the concern which shows that the framework architecture presents a non-risk for the tested quality attributes.

7.2.1 Interoperability Evaluation Scenarios

Table 7.1 presents the main features of the framework that provide *interoperability* enhancement, the table is a summary for the scenarios (questions and prompts) based on ATAM methodology that is presented in Appendix G.I.

Table 7.1: Framework Interoperability Supporting Features

#	Interoperability Supporting Features
1.	The framework supports diverse services implemented in various platforms and languages
2.	The framework allows replicating heterogeneous database types.
3.	The framework is using BPEL for business process which can orchestrate web services that uses SOAP and WSDL for service interfacing regardless of the underlying platform or development languages
4.	The framework allows having service users and providers to use different implementation languages and platforms.
5.	The middleware integration approach in the framework will be an ESB. It would allow connecting diverse applications, technologies, and data formatting.
6.	The authentication mechanism will be centralized and realized using Web Service.
7.	The framework is designed to support standard message-level security; but it is left for the service provides to implement such features to further enhance the security of the service.
8.	The ESB is responsible for integrating legacy systems to the framework,

#	Interoperability Supporting Features
	which provides interoperability with old legacy applications.
9.	The standards used in the framework and provide *interoperability* between framework components when interacting with each others are: WSDL, SOAP, UDDI, and BPEL which provide capabilities to systems developed with Web services technology.
10.	Not all Web services platforms implement the same version of the additional standards such as UDDI, BPEL, WS-Security, and hence achieving interoperability faces some obstacles when using such standards. Still since the framework is under a centralized unit of administration, this risk can be mitigated.

7.2.2 Manageability Evaluation Scenarios

Table 7.2 presents the main features of the framework that provide *manageability* enhancement, the table is a summary for the scenarios (questions and prompts) based on ATAM methodology that is presented in Appendix G.II.

Table 7.2: Framework Manageability Supporting Features

#	Manageability Supporting Features
1.	The ESB supports a centralized point of management of the services. The System Management Service also provides management capability for metric usage and health monitor of the framework services
2.	The Management Service in the framework provides the capability of accessing all logs related with the services usage, and provides presentation logic for the framework logs repository.
3.	The framework allows for an authentication that is centralized through using the Security Assurance Service, the authentication can be used by all services, and hence provide a central point of authentication management.
4.	The framework provides metric usage when using Security Assurance Service which records a log of the services access, in addition to this the ESB and the application server provides a metric usage and logging of services usage.

#	Manageability Supporting Features
5.	The BPEL engine manages BPEL processes and the application server that runs the engine in its context provides monitoring and logging of event data and measurement of business metrics such as wait time, transaction volumes, and exception counts.
6.	The framework through the ESB and application server provides a monitoring facility for the invoked services, and the health of the framework components.
7.	The ESB and the application server under which the services run allow starting and stopping framework engines, components, and services.
8.	The mechanisms for monitoring and event logging allow taking measures, such as wait times, transaction volumes, and exception counts. These measures are important to oversee the system in production and for the testing of reliability and performance analysis.

7.2.3 Flexibility Evaluation Scenarios

Table 7.3 presents the main features of the framework that provide *flexibility* enhancement, the table is a summary for the scenarios (questions and prompts) based on ATAM methodology that is presented in Appendix G.III.

Table 7.3: Framework Flexibility Supporting Features

#	Flexibility Supporting Features
1.	Since the framework is composed of diverse components, most of them are Web Services, which are self-contained and loosely coupled; the changes required for any service would be incur little efforts.
2.	Services in the process can be changed without affecting every other service in the BPEL workflow, as far as the input/output types of the service are not changed.
3.	The identity information is not hard-coded in security services implementation. Depending on the realization of the Web Service, access information can be stored in an XML file or database data source.
4.	The Web Services to be implemented are coarse grain and hence self

#	Flexibility Supporting Features
	contained and can operate independently, so they provide loose coupling and enhances flexibility.
5.	The credentials are used in Security Assurance Service; they are easily managed and not hard coded. They are stored in an XML format, which is flexible for manipulating and easy for understanding.
6.	Access to services by consumers can be from the Internet as an open and insecure network, as well as, from the private governmental network.
7.	Using a new data source in the framework, or adding another database type can be achieved with little efforts, because a minimal change is required in the code that access the database, if such database connectivity is not supported by the JDBC.
8.	The framework provides support for both synchronous and asynchronous web service. It is left for the requirement and operation of the service to use either of them. The services implemented in the prototype are synchronous services.

7.3 Showing Quality Attributes Achievement through a Usage Scenario

To further illustrate the idea presented in this section; we consider a usage scenario of the implemented prototype of framework. Figure 7.2 depicts the flow of this scenario and the interaction direction between different components, as follows:

1. The end user –e.g. citizen- would like to check his social status in the Citizen Population Registry, for example, in order to verify that his new born baby was added to the social section part of his identification card, and over the Internet, accesses the citizen information section of e-Government portal using his login credential.
2. The Web application running at the portal would use SOAP messaging and HTTP to invoke ,say, the *Citizen Information Operation* which is part of the *Government Informational Services - CtzDataService*

3. The *Government Informational Service* would then interact with the *Security Assurance Service* to make sure security policies are not violated and access for this context is allowed.
4. The *Security Assurance Service* and based on username/password pairs and IP authentication would allow the *Government Informational Service.*
5. The *Government Informational Service* would invoke the *Database Access Service* for information retrieval from the database.
6. *The Database Access Service* accesses the Central Database using the JBI database binding component or database driver.
7. The Database returns the requested records to the *Database Access Service.*
8. The *Database Access Service* database returns the response to the *Government Information Service.*
9. The *Governmental Information Service* manipulates and processes the results and return it back to the Web application at e-Government portal
10. Finally, the Web application renders, formats, and presents the required information to the citizen.

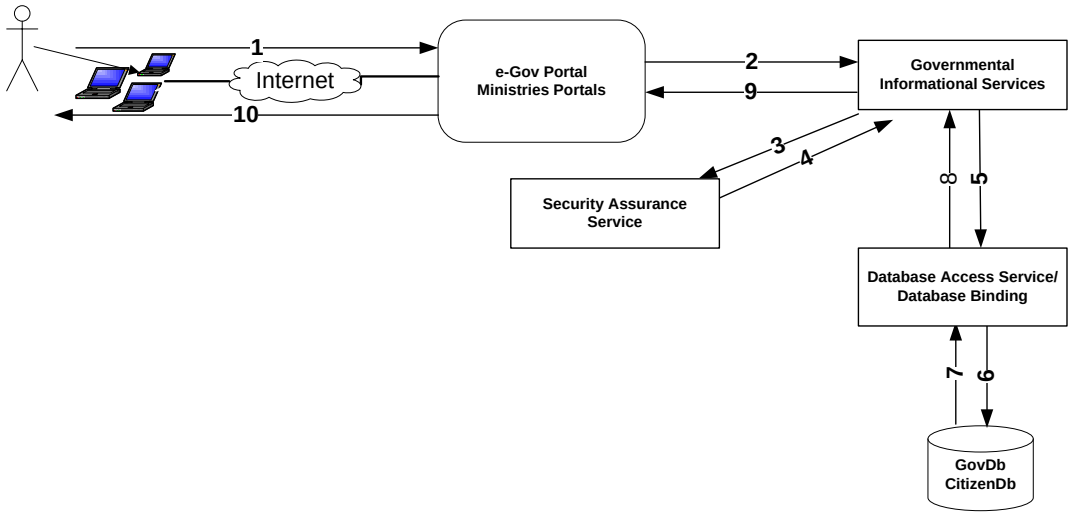

Figure 7.2: Usage Scenario for the Proposed Framework

The presented scenario outlines how the implemented prototype fulfills one of the main functional requirements, namely the accessibility, as well as two of the quality attributes which are *interoperability* and *flexibility*.

For the quality attributes discussion, first, *interoperability* achievement is clear in this scenario, this is because the Web application and the *Government Informational Service* are database type independent, and so if the low level database that holds the Citizens

Population Registry is changed from e.g. Oracle to MySQL, then change is not required for the Web application that access the Governmental Database. Second, *flexibility* is achieved by using standard HTTP transport to carry messages between the Web application in e-Government portal and the *Government Informational Service* (either over the government private network or the Internet); the HTTP transport is generally allowed and not filtered by firewalls; where in the current model of Central Database such access would be carried over Oracle-Sql port which most of the time needs security reconfiguration to allow it, also the portal access to the Central Database is restricted to be from the government private network.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this research, the current Integrated Central Database model, a core part of the Palestinian e-Government Technical Framework, was presented and analyzed. A new Central Database model based on SOA solution was proposed that overcomes the shortcomings of the currently used model that lacks Interoperability, Flexibility and Manageability. The proposed framework is based on SOA hub and spoke approach and was realized using an ESB and Web Services. The framework provides the main functionalities which are the access to the Integrated Central Database, and the replication between the diverse database types. The framework structure and components were presented and explained. The main components of the framework are: ESB, Web Services, databases, e-Government portals, business applications of governmental institutes, and front-end applications.

In the evaluation of the thesis work, we used two methods: ATAM based and proof-of-concept. The ATAM based evaluation method was used to evaluate the framework architecture, in which we introduced different scenarios that affect the quality attributes of the framework. The scenarios were presented using questions and their prompts, and the prompts conclude and verify that the framework achieves the quality goals. A prototype was implemented for limited functions of the framework. The implementation included a usage case of three informational services, replication service, security assurance service, and service orchestration using BPEL for the informational services. The development environment of the prototype was based on JSP, Netbeans, JBI and Open-ESB, and the replication between the database Oracle and MySQL was achieved using database binding component which is part of the JBI. The prototype was the proof-of-concept to validate the solution of the framework and showed it accomplishes its requirements.

The main contribution and impact of this research is to show that SOA solutions can be applied to the Integrated Central Database model, also to align SOA concepts to the e-Government domain problems.

8.2 Future Work

In this research we focused on building a SOA based framework for the e-Government Integrated Central Database as a model to replace the legacy one. We evaluated the framework architecture and validated the solution using a prototype. Yet, the complete framework was not implemented. Also the Framework did not address features that can further enhance flexibility and dependability such as service auto-composition. Moreover the framework did not address the issue of distributed Governmental Database to provide fault tolerance and reliability. In addition to this, services finding and invocation in the framework did not address semantic approach. Future direction in this research could be summarized as follows:

- Full and complete implementation of the framework.
- Enhancing the framework by adding support of services auto-composition.
- Providing semantic capabilities to the framework.
- Adding support for integrated and distributed database backend, instead of having just one Integrated Centralized Governmental Database.
- Enhancing the framework to support and achieve the following quality attributes: availability, reliability and fault-tolerance.

References

[1] Abdul Moiz S., Sailaja P., Venkataswamy G., and Pal S, "Database Replication: A Survey of Open Source and Commercial Tools", International Journal of Computer Applications, Published by Foundation of Computer Science, Jan. 2011.

[2] Alonso G., Casati, F., Kuno H. and Machiraju V., "Web Services: Concepts Architecture and Applications", Springer, Berlin, 2004.

[3] Apache ServiceMix Open Source ESB, http://servicemix.apache.org/home.html, Last Accessed 09/10/2011.

[4] Bansiya J., and Davis C.G., "A hierarchical Model for Object-Oriented Design Quality Assessment", IEEE Transactions on Software Engineering, 2002.

[5] Barbacci M., Clements P., Lattanze A., Northrop L., and Wood W., "Using the Architecture Tradeoff Analysis Method (ATAM) to Evaluate the Software Architecture for a Product Line of Avionics Systems: A Case Study (CMU/SEI-2003-TN-012, ADA418415)", Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[6] Barret T., "A Gentle Introduction to GlassFish ESB", Sun Microsystems Inc., 2009.

[7] Bass L., Clements P. and Kazman R., "Software Architecture in Practice", 2nd ed. Boston, MA: Addison-Wesley, (ISBN 0-321-15495-9), 2003.

[8] Bianco P., Kotermanski R., and Merson P., "Evaluating a Service Oriented Architecture", Engineering Institute, 2007.

[9] Biemolt G., and Groefsema H., "Replicating Subsets of Data for the Dutch E-Government", Technical Report, University of Groningen and Ordina Oracle Solutions, 2008.

[10] Bih J., "Service Oriented Architecture (SOA) A New Paradigm to Implement Dynamic E-business Solutions", Ubiquity ACM, New York, NY, USA. August 2006.

[11] Changyu Z., Jianyong D., and Zheng X., "Beijing Traffic Data Center Based on SOA Technology", International Conference on Computer Application and System Modeling (ICCASM), vol.15, pp.232-235, 22-24 Oct. 2010.

[12] Chappell D., "Enterprise Service Bus Theory in Practice", O'Reilly, 2004.

[13] Chen D., "Enterprise Interoperability Framework", In Proceedings of Enterprise

Modeling and Ontologies for Interoperability EMOI - Interop, vol. 200, 2006.

[14] Correia A. J., Pereira J. O., Rodrigues L., Carvalho N. M. R., Vila_ca R., Oliveira R., and Guedes S., "An open architecture for database replication", In Proc. of the 6th International Symposium on Network Computing and Applications, Boston, MA, USA IEEE, pages 287-290, , July 2007.

[15] Curl A., and Fertalj K., "A review of enterprise IT Integration Methods", Proceedings of the 31st International Conference on Information Technology Interfaces (ITI '09), Dubrovnik, pp.107-112, 22-25 June 2009.

[16] Dan A., Johnson R., and Rsanjani. A, "Information as a Service: Modeling and Realization", International Workshop on Systems Development in SOA Environments, 2007. SDSOA '07: ICSE Workshops 2007, pp. 2, 20-26 May 2007.

[17] Delin Q., "Design of Medical Insurance Supervision System Based on Active Data Warehouse and SOA", World Congress on Computer Science and Information Engineering WRI, vol.3, pp.45-49, March 31-April 2 2009.

[18] ESB Tutorial, http://searchsoa.techtarget.com/tutorial/ESB-Tutorial, Last Accessed 5/10/2011.

[19] Fiere J., "SOA Security", Master's thesis, Faculty of Science, Vrije University Amsterdam, 2007.

[20] Garcia-Jimeenez F.J., Martinez-Carreras M.A.,and Gomez-Skarmeta A.F., "Evaluating Open Source Enterprise Service Bus", IEEE 7th International Conference on e-Business Engineering ICEBE, pp.284-291, 2010.

[21] Guo Q., Zheng H., Lie J., and Wang X., "Design and Implementation of Data Management Center Based on Web Services", Ninth International Conference on Hybrid Intelligent Systems HIS 09, vol.2, pp.322-326, 12-14 Aug. 2009.

[22] Gupta A., and Mumick I. S., "Maintenance of Materialized Views: Problems, Techniques, and Applications", Data Engineering Bulletin, vol. 18, No. 2, June 1995.

[23] IBM WebSphere ESB, http://www-01.ibm.com/software/integration/wsesb/, Last Accessed 5/10/2011.

[24] IEEE Standard Glossary of Software Engineering Terminology 610.12-1990, vol. 1, Los Alamitos: IEEE Press, 1999.

[25] Introducing OpenESB from development to Administration and Management, http://oracamp.com/article-introducing-openesb-development-administration-and-management, Last Accessed 09/10/2011.

[26] Introduction to Oracle Enterprise Service Bus, http://download.oracle.com/docs/cd/B31017_01/integrate.1013/b28211/esb_intro.htm, Last Accessed 25/9/2011.

[27] JBoss ESB, http://www.jboss.org/jbossesb, Last Accessed 09/10/2011.

[28] Jones G. L., and Lattanze J. A., "Using the Architecture Tradeoff Analysis Method to Evaluate a Wargame Simulation System: A Case Study (CMU/SEI-2001-TN-022, ADA399795). Pittsburgh, PA", Software Engineering Institute, Carnegie Mellon University, 2001.

[29] Josuttis M. N., "SOA in practice", O'Reilly Media Inc., 2007.

[30] Kazman R., Klein M., and Clements P., "ATAM: Method for Architecture Evaluation", August 2000.

[31] Keen M., Bishop S., Hopkins A., Milinski S., NottRick C., Robinson R., Adams J., Verschueren P., and Acharya A., "Patterns: Implementing an SOA using an Enterprise Service Bus", IBM Redbook, 2005.

[32] Kemme B., Alonso G., "Database Replication: a Tale of Research across Communities", Journal Proceedings of the VLDB Endowment, vol. 3, issue 1-2, September 2010.

[33] Krolczyk A., Stantchev V., and Senf C., "Service-Oriented Approaches for E-Government", Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services IIWAS 09, ACM, New York, NY, USA, 2009.

[34] Kuppuraju S., Kumar A., Kumari G.P., "Case Study to Verify the Interoperability of a Service Oriented Architecture Stack", IEEE International Conference on Services Computing SCC, pp. 678-679, 9-13 July 2007.

[35] Lee Y., "QAM: QoS-assured Management for Interoperability and Manageability for SOA", Second Pacific-Asia Conference on Circuits, Communications and System (PACCS), vol.1, pp.468-472, 1-2 Aug. 2010.

[36] Lublinsky B., "An ESB Vendors Evaluation by Forrester Research", http://www.infoq.com/news/2009/02/ESBVendors, 2009.

[37] Ma H., "A Service-oriented e-Government Support Platform for Integration of Application and Data", Second International Conference on Information Technology and Computer Science (ITCS), pp.398-401, 24-25 July 2010.

[38] MacKenzie M., Laskey K., McCabe F., Brown P., and Metz. R., "Reference Model for Service Oriented Architecture", OASIS Committee, February 2006.

[39] Mann T. A., ".NET Web Services for Dummies", New York, Wiley Publishing, Inc., 2003.

[40] Mar O., "Fault Tolerance by Replication of Distributed Database in P2P System using Agent Approach", International Journal of Computers, issue 1. vol.4, 2010.

[41] Materialized View Concepts and Architecture, http://download.oracle.com/docs/cd/B10500_01/server.920/a96567/repmview.htm,

Last Accessed 15/9/2011.

[42] Medjahed B., Rezgui A., Bouguettaya A., and Ouzzani M., "Infrastructure for e-government Web Services", Internet Computing IEEE, vol.7, no.1, pp. 58- 65, Jan/Feb 2003.

[43] Meier J.D. , Hill D., Homer A., Taylor J., Bansode P., Wall L., Boucher R., and Bogawat A., "Microsoft Application Architecture Guide 2.0a, Patterns and Practices", Microsoft Press, 2nd edition, 2009.

[44] Menge F., "Enterprise Service Bus", Free and Open Source Software Conference, Vicrtoria, Canda, 2007.

[45] Michelson B., "Enterprise Service Bus Q&A (Part II of II)", http://www.ebizq.net/topics/esb/features/6117.html, Last Accessed 09/10/2011.

[46] Minguez J., Jakob M., Heinkel U., and Mitschang B., "A SOA-based Approach for the Integration of a Data Propagation System", International Conference on Information Reuse & Integration IRI 09. IEEE, pp. 47-52, 10-12 Aug. 2009.

[47] Ministry of Telecom and Information Technology –Gaza - Palestine- Official Web Site, http://www.mtit.gov.ps, Last Accessed 05/08/2011.

[48] Mule ESB Open Source ESB Community, http://www.mulesoft.org/, Last Accessed 09/10/2011.

[49] Open Source SOA- Fuse Open Source Community, http://fusesource.com/, Last Accessed 25/9/2011.

[50] OpenESB, http://java.net/projects/openesb/, Last Accessed 09/10/2011.

[51] Ortiz S., "Getting on Board the Enterprise Service Bus", IEEE Computer Magazine, vol.40, issue 4, pp.15-17, April 2007.

[52] Palestinian Government Data Integration Committee Official Website, http://www.takamul.gov.ps, Last Accessed 05/10/2011.

[53] Papazoglou M., "Service-oriented Computing: Concepts, Characteristics and Directions", Proceedings of the Fourth International Conference on Web Information Systems Engineering WISE 2003, pp. 3- 12, 10-12 Dec. 2003.

[54] Papazoglou M., and Heuvel W., "Service Oriented Architectures: Approaches, Technologies and Research Issues", Journal on Very Large Data Bases, 2007.

[55] Papazoglou M., Traverso P., Dustdar S. and Leymann F., "Service-Oriented Computing Research Roadmap", International Journal of Cooperative Information Systems, vol. 17 no. 2, p223-255, 2008

[56] Papazoglou M., Traverso P., Dustdar S., and Leymann F., "Service-Oriented Computing Research Roadmap", International Journal of Cooperative Information Systems, vol. 17 no. 2, p223-255, 2008.

[57] Papazoglou M., Traverso P., Dustdar S., and Leymann F., "Service-Oriented Computing: State of the Art and Research Challenges", International Journal of Cooperative Information Systems, vol. 17, no. 2, 2008.

[58] Paul S., "Pro SQL Server 2008 Replication", Apress, 1st edition June 17th 2009.

[59] Petals ESB, Open source ESB, http://petals.ow2.org/, Last Accessed 09/10/2011.

[60] Philipp L., "The Strategic Impact of Service Oriented Architectures", 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07), pp. 475-484, 2007.

[61] PostgreSQL 8.4 Server Administration, Volume II- The PostgreSQL Global Development Group, 2009.

[62] Proof of Concept, http://en.wikipedia.org/wiki/Proof_of_concept, Last Accessed 25/9/2011.

[63] RadeMakers T. and Dirksen J., "Open Source ESB in Action", Manning Publications, 2008.

[64] Rantzau R., Constantinescu C., Heinkel U., and Meinecke H., "Champagne: Data Change Propagation for Heterogeneous Information Systems", Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[65] Saito Y., and Shapiro M., "Replication: Optimistic Approaches", Hewlett-Packard Laboratories Palo Alto, Microsoft Research Ltd. Cambridge, 2002.

[66] Sarev V. G., "Process and Realization of SOA Centralized System", Master's thesis, University of Sofia "St Kliment Ohridski" Faculty of Mathematics and Informatics Department: Information technologies, 2007.

[67] Simmons S., Designing SOA with a business focus, IBM WebSphere Developer Technical Journal, 2007, available at: http://www.ibm.com/developerworks/websphere/techjournal/0706_col_simmons/0706_col_simmons.html.

[68] SOA and Web Services— The Performance Paradox, White Paper, Wily Technology, August 2007.

[69] Strahle M., Ehlbeck M., Prapavat V., Kuck K., Franz F., and Meyer J.-U., "Towards a Service-Oriented Architecture for Interconnecting Medical Devices and Applications", Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability HCMDSS-MDPnP., pp.153-155, 25-27 June 2007.

[70] TIBCO ActiveMatrix BusinessWorks, http://www.tibco.com/software/soa/activematrixbusinessworks/, Last Accessed 09/10/2011.

[71] Wagner R. and Mitschang B., "Uniform and Efficient Data Provisioning for SOA-

Based Information Systems", Sixth International Conference on Information Technology: New Generations ITNG '09. , pp.1012-1017, 27-29 April 2009.

[72] Wang J., Yu A., Zhang X., and Qu L., "A Dynamic Data Integration Model Based on SOA", International Colloquium on Computing, Communication, Control, and Management CCCM, vol.2, pp.196-199, 8-9 Aug. 2009.

[73] Web Services Architecture, W3C Working Group Note, Web Services Definition, http://www.w3.org/TR/ws-arch/#whatis, Last Accessed 5/8/2011.

[74] Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[75] Yang C., Huang G., Huang F., Ho T., Huang P., and Jiann-Min Y., "SOA-based Platform for Water Resource Information Exchanging", 17th International Conference on Geoinformatics, pp.1-4, 12-14 Aug. 2009.

[76] Yunliang J., Xiongtao Z., Qing S., Jing F., and Ning Z., "Design of E-Government Information Management Platform Based on SOA Framework", First International Conference on Networking and Distributed Computing ICNDC , pp.165-169, 21-24 Oct. 2010.

[77] Zimmermann O., Tomlinson R. M., and Peuser S., "Perspectives on Web Services – Applying SOAP, WSDL and UDDI to Real- World Projects", Springer Professional Computing, 2005.

Appendices

These appendices provide an overview of prototype and excerpts and snapshots from the development environment, XSD, WSDL files and the part of the implementation codes. Also includes the scenarios for the evaluation of the framework.

The appendices included in this thesis are:

- Appendix A: Prototype Working Environment
- Appendix B: Informational Web Services
- Appendix C: The Informational Services Composite Applications and BPEL
- Appendix D: Security Service Assurance Implementation Files
- Appendix E: Replication Service, Database Binding, and BPEL
- Appendix F: Front End Access Interface
- Appendix G: Framework Evaluation Scenarios based on ATAM

Appendix A: Prototype Working Environment

- The prototype has been implemented using Java Web Services; Table A.1 shows the used software for the implementation of the prototype.

Table A.1 Software Used in the Implementation of Prototype

Software Function	Software Environment
Java Development Kit	JDK 1.6 jdk1.6.0_05
Development Environment	NetBeans IDE 6.7.1
Application Server	GlassFish Application Server 2.1
ESB	OpenESB/JBI jbi_components_installer.jar
ESB Engines and Binding Components	Open_esb_v2 jbi_components_installer.jar sub-bpel-engine sun-database-binding
Database	MySQL Database Oracle 10g Database
Database Connectivity	JDBC/ojdbc6.jar mysql-connector-java-5.1.16-bin.jar

- Figure A.1 depicts a snapshot of the development environment which is NetBeans IDE 6.71.

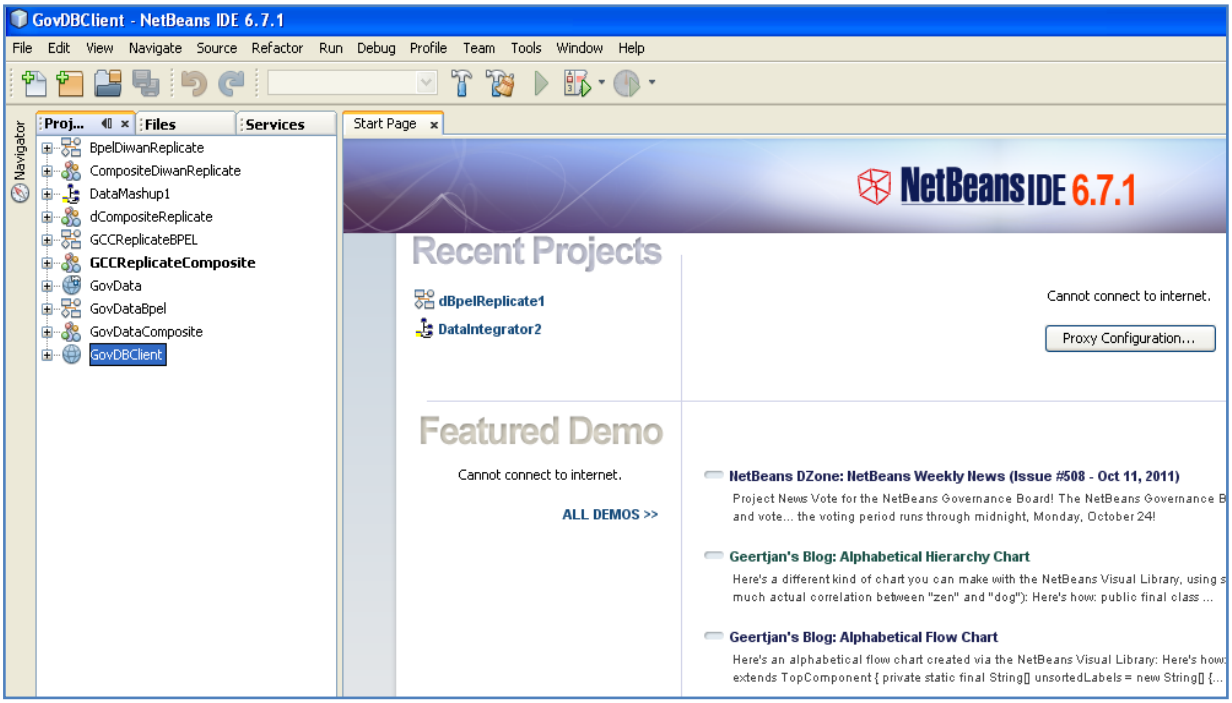

Figure A.1: IDE Snapshot

Appendix B: Informational Web Services

The GovData Web Application holds the Informational Web Services as well as the Security Assurance Web Service. Figure A.2 shows the Web Services and their operations.

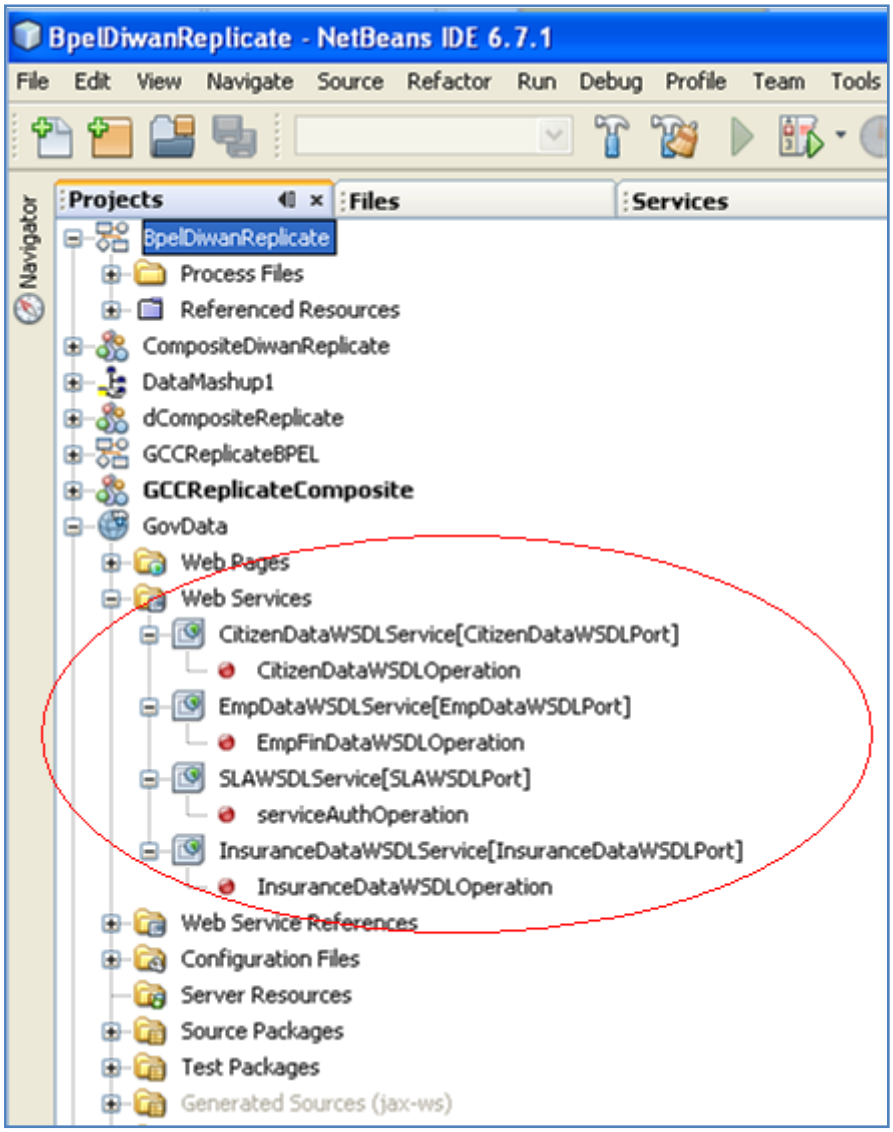

Figure A.2: Informational and Security Web Services and their Operations

Following the required files for the implementation of the *CitizenData* Web Service are presented.

- XSD file for CitizenData Service

Figure A.3 depicts the Schema XSD file for the CitizenData Web Service

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="h
    elementFormDefault="qualified">
<xsd:complexType name="CitizenDataInType">
   <xsd:sequence>
     <xsd:element name="username" type="xsd:string" />
     <xsd:element name="password" type="xsd:string" />
     <xsd:element name="id" type="xsd:int" />
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CitizenDataOutType">
  <xsd:sequence>
    <xsd:element name="citizenID" type="xsd:int" />
       <xsd:element name="firstName" type="xsd:string" />
       <xsd:element name="secondName" type="xsd:string" />
       <xsd:element name="thirdName" type="xsd:string" />
       <xsd:element name="lastName" type="xsd:string" />
       <xsd:element name="englishName" type="xsd:string" />
       <xsd:element name="motherName" type="xsd:string" />
       <xsd:element name="prevLastName" type="xsd:string" />
       <xsd:element name="iGender" type="xsd:int" />
       <xsd:element name="sGender" type="xsd:string" />
       <xsd:element name="iReligion" type="xsd:int" />
       <xsd:element name="sReligion" type="xsd:string" />
       <xsd:element name="iMaritalStatus" type="xsd:int" />
       <xsd:element name="sMaritalStatus" type="xsd:string" />
       <xsd:element name="birthDate" type="xsd:string" />
       <xsd:element name="birthPlace" type="xsd:string" />
       <xsd:element name="sRegion" type="xsd:string" />
       <xsd:element name="iRegion" type="xsd:int" />
       <xsd:element name="sCity" type="xsd:string" />
       <xsd:element name="iCity" type="xsd:int" />
       <xsd:element name="street" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="CitizenDataRequest" type="tns:CitizenDataInType"/>
<xsd:element name="CitizenDataRecord" type="tns:CitizenDataOutType"/>
<xsd:element name="CitizenDataFault" type="xsd:string"></xsd:element>
</xsd:schema>
```

Figure A.3: CitizenData Web Service XSD Schema File

- WSDL file for the CitizenData Service

Figure A.4 shows the WSDL file the CitizenData Web Service, it has one operation, three messages (input, output, and fault).

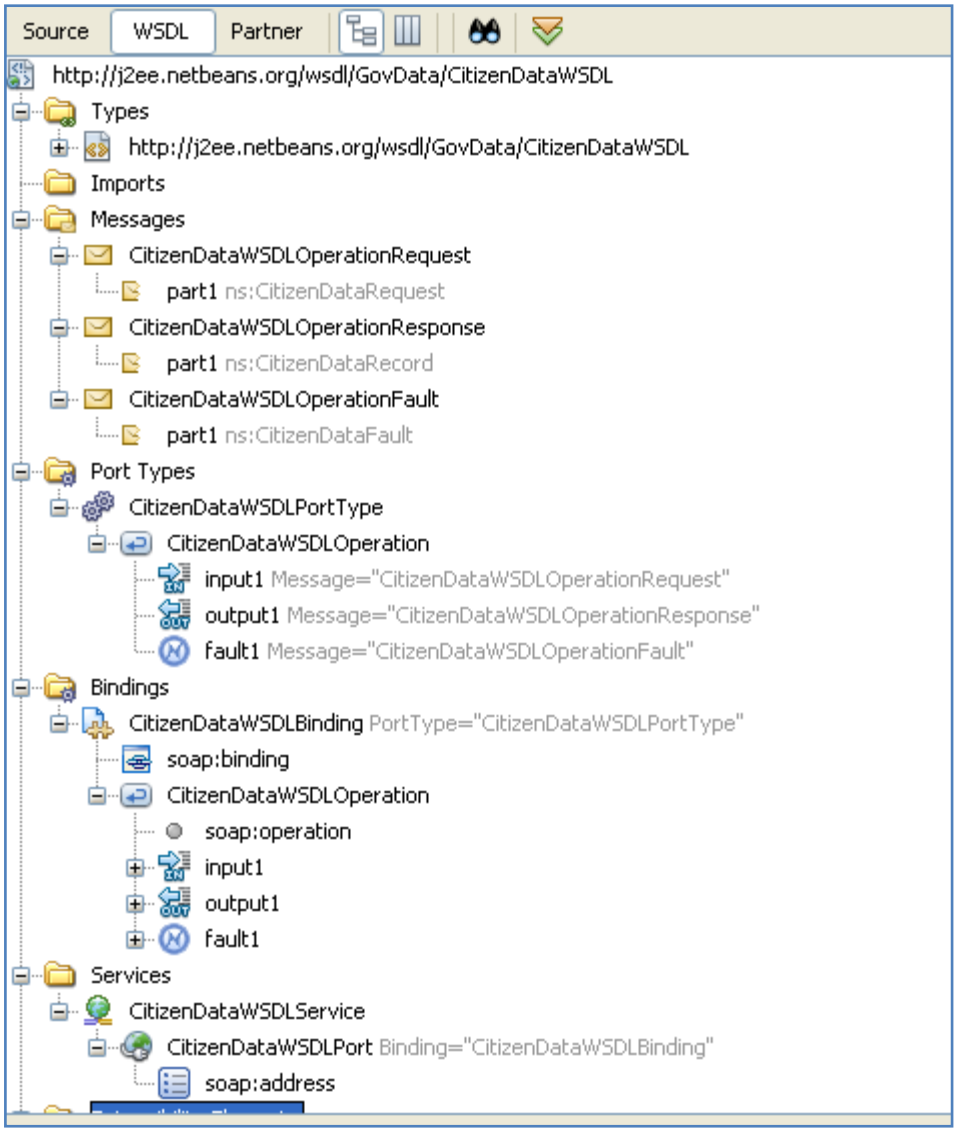

Figure A.4: CitizenData Web Service WSDL

- CitizenData Web Service Implementation

Figure A.5 depicts the Java code for the CitizenData Web Service

```
package ws;
import javax.jws.WebService;
import org.netbeans.j2ee.wsdl.govdata.citizendatawsdl.CitizenDataWSDLOperationFault;
import java.sql.*;
import javax.annotation.Resource;
import javax.jws.WebMethod;
import javax.servlet.http.HttpServletRequest;
import javax.xml.ws.handler.MessageContext;
import javax.xml.ws.WebServiceContext;
import oracle.jdbc.*;
import oracle.sql.CharacterSet.*;
@WebService(serviceName = "CitizenDataWSDLService", portName = "CitizenDataWSDLPort",
endpointInterface =
"org.netbeans.j2ee.wsdl.govdata.citizendatawsdl.CitizenDataWSDLPortType",
targetNamespace = "http://j2ee.netbeans.org/wsdl/GovData/CitizenDataWSDL",
wsdlLocation = "WEB-INF/wsdl/CitizenDataWSDLService/CitizenDataWSDL.wsdl")
public class CitizenDataWSDLService {
@Resource
WebServiceContext wsContext;
WebMethod wMethod;

    public org.netbeans.xml.schema.citizendataschema.CitizenDataOutType
       citizenDataWSDLOperation(org.netbeans.xml.schema.citizendataschema.CitizenData
InType part1) throws CitizenDataWSDLOperationFault {

 org.netbeans.xml.schema.citizendataschema.CitizenDataOutType Result =
 new org.netbeans.xml.schema.citizendataschema.CitizenDataOutType();
        try {
             org.netbeans.j2ee.wsdl.govdata.slawsdl.SLAWSDLService slaService =
                      new  org.netbeans.j2ee.wsdl.govdata.slawsdl.SLAWSDLService();
              org.netbeans.j2ee.wsdl.govdata.slawsdl.SLAWSDLPortType slaPort =
slaService.getSLAWSDLPort();
              MessageContext mc = wsContext.getMessageContext();
              HttpServletRequest req =
(HttpServletRequest)mc.get(MessageContext.SERVLET_REQUEST);
              //check SLA service for user access rights
                String username= part1.getUsername();
                String password=part1.getPassword();
                String ipAddr = req.getRemoteAddr();
                String methodName =
Thread.currentThread().getStackTrace()[1].getMethodName();
                String serviceURI = req.getServletPath() +"/"+ methodName ;
                org.netbeans.xml.schema.slaschema.ServiceAuthInType authInRec =
                           new
org.netbeans.xml.schema.slaschema.ServiceAuthInType();
                authInRec.setIpAddr(ipAddr);
                authInRec.setUsername(username);
                authInRec.setPassword(password);
                authInRec.setServiceURI(serviceURI);
                authInRec.setNotes(Integer.toString(part1.getId()));
                try {
                     slaPort.serviceAuthOperation(authInRec);
                } catch  ( Exception ex)
                   {
                    System.out.println("Error in calling authentication service");
```

Figure A.5 (Part 1): Java Code for CitizenData Web Service

```
                    throw new CitizenDataWSDLOperationFault("Error SLAWSDLService
"+ex.getLocalizedMessage(),null);
                   }
            String jdbcURL = "jdbc:oracle:thin:@127.0.0.1:1521:govdata";
            Connection conn = null;
            OracleCallableStatement stmt;
            String dbUser = "gov_admin";
            String dbPasswd = "gov_admin";
            try {
               Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
               conn = DriverManager.getConnection(jdbcURL, dbUser, dbPasswd);
            }catch (ClassNotFoundException ex) {
                throw ex;
            }
            stmt = (OracleCallableStatement) conn.prepareCall("{call
MOI_GENERAL_PKG.CITZN_INFO(?,?)}");
            stmt.setInt(1, part1.getId());
            stmt.registerOutParameter(2, OracleTypes.CURSOR);
            stmt.execute();
            ResultSet rs = (ResultSet) stmt.getCursor(2);
            int count =0;
            while (rs.next()) {
                count++;
                Result.setCitizenID(part1.getId());
                Result.setFirstName(rs.getString(2));
                Result.setSecondName(rs.getString(3));
                Result.setThirdName(rs.getString(4));
                Result.setLastName(rs.getString(5));
                Result.setBirthDate(rs.getString(10));
                Result.setSGender(rs.getString(19));
                Result.setSMaritalStatus(rs.getString(20));
                Result.setSRegion(rs.getString(11));
                Result.setSCity(rs.getString(21));
            }
          if ( count == 0){
              throw new CitizenDataWSDLOperationFault("Invalid ID",null);
          }

        } catch (Exception  ex) {
            throw new CitizenDataWSDLOperationFault("ERROR
CitizenDataWSDLService:CitizenDataOperation: "+
                    ex.getLocalizedMessage(),null);
        }
  return Result;
    }
    }
```

Figure A.5 (Part 2): Java Code for CitizenData Web Service

Appendix C: The Informational Services Composite Application and BPEL

- Composite Service for the Informational Services

Figure A.6 depicts the WSDL file for the composite service that orchestrates the two Informational Services: CitizenData and InsuranceData Web Services.

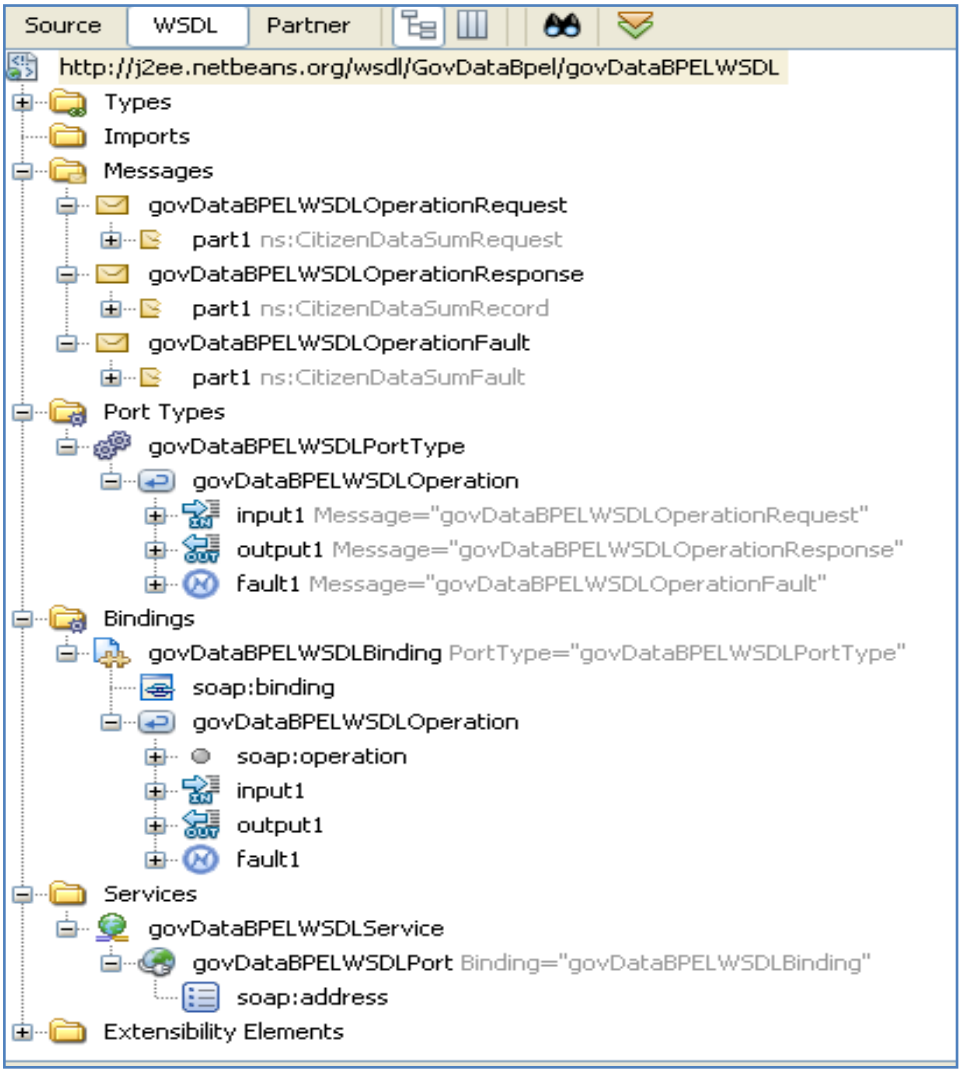

Figure A.6: WSDL File for the Informational Composite Service

- Composite Informational Services BPEL Design View:

In this view the two Informational Services (*CitizenData* and *InsuranceData*) are to right of Figure A.7, and the composite Web Service to the left.

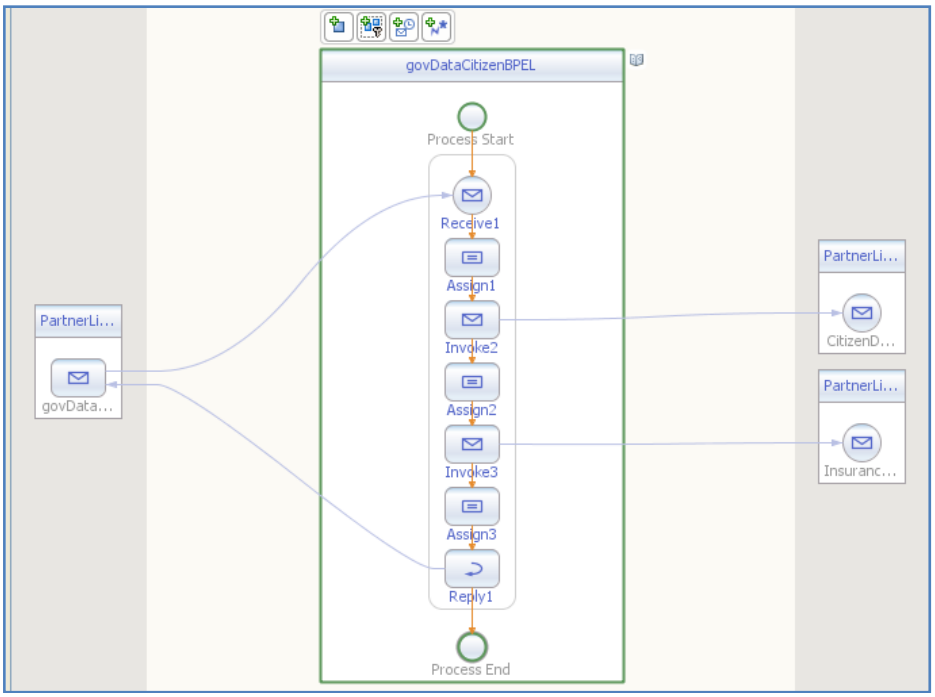

Figure A.7: BPEL Design for Informational Composite Web Services

Appendix D: Security Service Assurance Implementation Files

- Figure A.8 depicts the XSD file for Security Assurance Service, which provides the Authentication Input Request for input message of the service.

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

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://xml.netbeans.org/schema/SLASchema"
    xmlns:tns="http://xml.netbeans.org/schema/SLASchema"
    elementFormDefault="qualified">

    <xsd:complexType name="serviceAuthInType">
    <xsd:sequence>
        <xsd:element name="ipAddr" type="xsd:string" />
        <xsd:element name="username" type="xsd:string" />
        <xsd:element name="password" type="xsd:string" />
        <xsd:element name="serviceURI" type="xsd:string" />
        <xsd:element name="notes" type="xsd:string" />
    </xsd:sequence>
</xsd:complexType>
    <xsd:element name="serviceAuthIn" type="tns:serviceAuthInType"></xsd:element>
    <xsd:element name="serviceAuthFault" type="xsd:string"></xsd:element>
    <xsd:element name="sessionID" type="xsd:string"></xsd:element>
</xsd:schema>
```

Figure A.8: XSD for the Security Assurance Service

- Figure A.9 depicts the WSDL file for Security Assurance Service

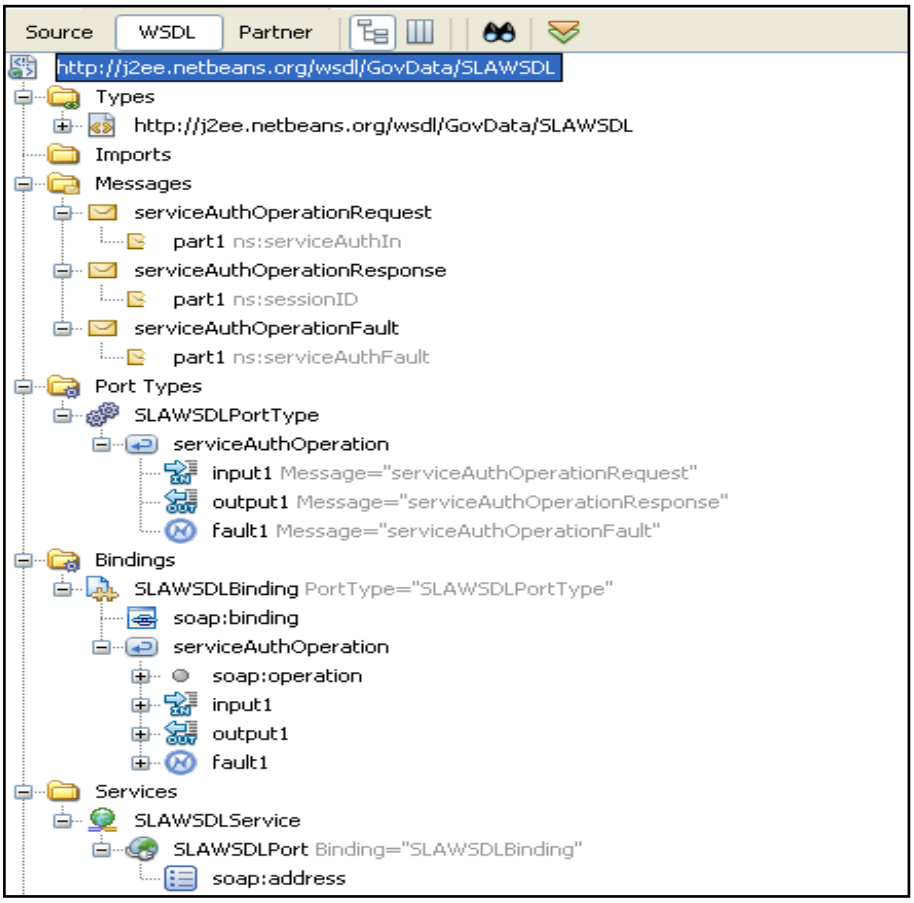

Figure A.9: WSDL File for the Security Assurance Service

- Security Assurance Authentication, Authorization, and Logging XML Files:

Shown in Figure A.10, A.11, and A.12 a snapshot of the XML files used by Security Assurance Service to authenticate, authorize, and log users access to operations in the services, as well as an excerpt from authorization code.

- users.xml file for authentication: An XML file showing username/password pair.

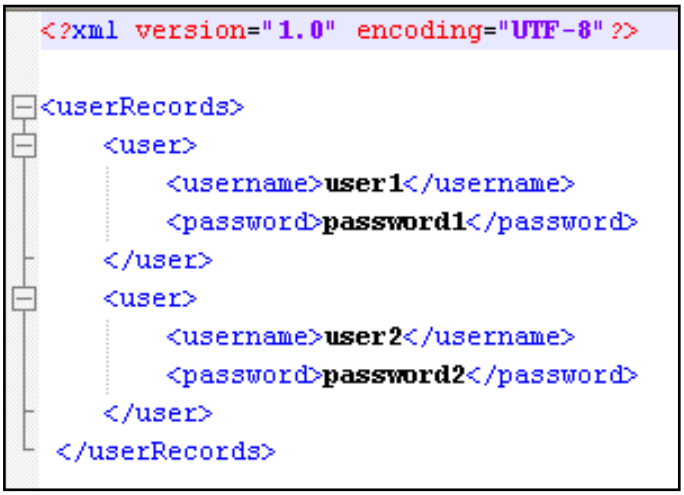

Figure A.10: Authorization XML File

- log.txt file for logging services requests: Each user request to operation is logged along with its access time.

```
Success 01-10-2011 21:19:46 user1  127.0.0.1  /CitizenDataWSDLService/citizenDataWSDLOperation 909521973
Success 01-10-2011 21:19:47 user1  127.0.0.1  /InsuranceDataWSDLService/insuranceDataWSDLOperation 909521973
Success 01-10-2011 21:19:47 user1  127.0.0.1  /EmpDataWSDLService/empFinDataWSDLOperation 909521973
Success 13-10-2011 07:12:37 user1  127.0.0.1  /CitizenDataWSDLService/citizenDataWSDLOperation 909521973
Success 13-10-2011 07:12:37 user1  127.0.0.1  /InsuranceDataWSDLService/insuranceDataWSDLOperation 909521973
Success 13-10-2011 07:12:38 user1  127.0.0.1  /EmpDataWSDLService/empFinDataWSDLOperation 909521973
Success 15-10-2011 07:14:06 user1  127.0.0.1  /CitizenDataWSDLService/citizenDataWSDLOperation 909521973
Success 15-10-2011 07:14:06 user1  127.0.0.1  /InsuranceDataWSDLService/insuranceDataWSDLOperation 909521973
Success 15-10-2011 07:14:07 user1  127.0.0.1  /EmpDataWSDLService/empFinDataWSDLOperation 909521973
Success 15-10-2011 07:15:49 user1  127.0.0.1  /CitizenDataWSDLService/citizenDataWSDLOperation 909521973
Success 15-10-2011 07:16:10 user1  127.0.0.1  /InsuranceDataWSDLService/insuranceDataWSDLOperation 909521973
Success 15-10-2011 07:17:42 user1  127.0.0.1  /EmpDataWSDLService/empFinDataWSDLOperation 909521973
Success 15-10-2011 07:23:23 user1  127.0.0.1  /CitizenDataWSDLService/citizenDataWSDLOperation 9091
Success 15-10-2011 07:23:24 user1  127.0.0.1  /InsuranceDataWSDLService/insuranceDataWSDLOperation 9091
Success 15-10-2011 07:24:05 user1  127.0.0.1  /EmpDataWSDLService/empFinDataWSDLOperation 9091
Success 15-10-2011 07:31:28 user1  127.0.0.1  /CitizenDataWSDLService/citizenDataWSDLOperation 120072098
Success 15-10-2011 07:31:28 user1  127.0.0.1  /InsuranceDataWSDLService/insuranceDataWSDLOperation 120072098
Success 15-10-2011 07:31:28 user1  127.0.0.1  /EmpDataWSDLService/empFinDataWSDLOperation 120072098
```

Figure A.11: Logging File

- access.xml for authorizing users for operations access: e.g. as seen from the figure below users1 is allowed to access the *empFinDataWSDLOperation* from a specific IP which is the loopback address 127.0.0.1.

```
<accessRecords>
    <acl>
        <username>user1</username>
        <ip>127.0.0.1</ip>
        <serviceUri>/EmpDataWSDLService/empFinDataWSDLOperation</serviceUri>
    </acl>
    <acl>
        <username>user1</username>
        <ip>127.0.0.1</ip>
        <serviceUri>/CitizenDataWSDLService/citizenDataWSDLOperation</serviceUri>
    </acl>
    <acl>
        <username>user1</username>
        <ip>127.0.0.1</ip>
        <serviceUri>/InsuranceDataWSDLService/insuranceDataWSDLOperation</serviceUri>
    </acl>
    <acl>
        <username>user1</username>
        <ip>192.168.0.1</ip>
        <serviceUri>/EmpDataWSDLService/empFinDataWSDLOperation</serviceUri>
    </acl>
    <acl>
        <username>user1</username>
        <ip>192.168.0.1</ip>
        <serviceUri>/CitizenDataWSDLService/citizenDataWSDLOperation</serviceUri>
    </acl>
    <acl>
        <username>user1</username>
        <ip>192.168.0.1</ip>
        <serviceUri>/InsuranceDataWSDLService/insuranceDataWSDLOperation</serviceUri>
    </acl>
</accessRecords>
```

Figure A.12: Authorization XML File

- Figure A.13 presents the Java code for Security Assurance Service, access to authentication, authorization and logging files is done using DOM-SAX.

```
package ws;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintStream;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.w3c.dom.*;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import javax.jws.WebService;
import org.netbeans.j2ee.wsdl.govdata.slawsdl.ServiceAuthOperationFault;
@WebService(serviceName = "SLAWSDLService", portName = "SLAWSDLPort", endpointInterface =
"org.netbeans.j2ee.wsdl.govdata.slawsdl.SLAWSDLPortType",
 targetNamespace = "http://j2ee.netbeans.org/wsdl/GovData/SLAWSDL", wsdlLocation = "WEB-
INF/wsdl/SLAWSDLService/SLAWSDL.wsdl")
public class SLAWSDLService {
public boolean authUsername(String user, String pass){
   File file = new File("users.xml");
   try {
     DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
     Document doc =   builder.parse(file);
     NodeList nodes = doc.getElementsByTagName("user");
     for (int i = 0; i < nodes.getLength(); i++) {
       Element element = (Element) nodes.item(i);
       NodeList username = element.getElementsByTagName("username");
       Element line = (Element) username.item(0);
       String usernameValue = line.getFirstChild().getNodeValue();
       NodeList password = element.getElementsByTagName("password");
       line = (Element) password.item(0);
       String passwordValue = line.getFirstChild().getNodeValue();

       if ( user.equals(usernameValue) && pass.equals(passwordValue)) {
                return true;
            }
    }
   }
   catch (Exception e) {
      System.out.println("Error reading users file "+e.getLocalizedMessage());
   }
return false;
}
 public boolean authUsernameIP(String user, String IP, String serviceURI){
   File file = new File("access.xml");
   try {
     DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
     Document doc =   builder.parse(file);
      NodeList nodes = doc.getElementsByTagName("acl");
     for (int i = 0; i < nodes.getLength(); i++) {
       Element element = (Element) nodes.item(i);
       NodeList username = element.getElementsByTagName("username");
       Element line = (Element) username.item(0);
       String usernameValue = line.getFirstChild().getNodeValue();
       NodeList ipAddr = element.getElementsByTagName("ip");
       line = (Element) ipAddr.item(0);
       String ipValue = line.getFirstChild().getNodeValue();
       NodeList serviceUri = element.getElementsByTagName("serviceUri");
       line = (Element) serviceUri.item(0);
       String serviceUriValue = line.getFirstChild().getNodeValue();
       if ( user.equals(usernameValue) && IP.equals(ipValue) &&
serviceURI.equals(serviceUriValue)) {
                return true;
            }
    }
     }
   catch (Exception e) {
```

Figure A.13 (Part 1): Security Assurance Java Code

```
System.out.println("Error reading users file "+e.getLocalizedMessage());
   }
return false;
}
 public boolean  isAuthenticatedService (String serviceURI){
        return true;
    }
 public void   logAuthRequest(String status,String username, String IP, String serviceURI,
String notes){
     Date todaysDate = new java.util.Date();
     SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");
     String formattedDate = formatter.format(todaysDate);
     FileOutputStream fout;
     try
             {
                 // Open an output stream
                 fout = new FileOutputStream ("log.txt",true);

                 // Print a line of text
                 new PrintStream(fout).println (status+" "+ formattedDate+" "+username+"
"+IP+"  "+serviceURI+" "+notes);
                 // Close our output stream
                 fout.close();
             }
             // Catches any error conditions
             catch (IOException e)
             {
                    System.err.println ("Unable to write to file"+e.getLocalizedMessage());
                    System.exit(-1);
             }
     System.out.println(status+" "+ formattedDate+" "+username+"  "+IP+"  "+serviceURI+"
"+notes);
    }
    public java.lang.String
serviceAuthOperation(org.netbeans.xml.schema.slaschema.ServiceAuthInType part1) throws
ServiceAuthOperationFault {
       try {
        String username= part1.getUsername();
        String password= part1.getPassword();
        String serviceURI= part1.getServiceURI();
        String notes = part1.getNotes();
        String ipAddr = part1.getIpAddr();
        if (! isAuthenticatedService(serviceURI))
          {
            return "ok";
          }
        if (authUsername(username, password)== false)
         {
            logAuthRequest("FailedLogin",username, ipAddr, serviceURI,notes);
            throw new ServiceAuthOperationFault("Invalid Username Password",null);
         }
        if ( authUsernameIP(username,ipAddr,serviceURI)== false)
         {
           logAuthRequest("AccessDenied",username, ipAddr, serviceURI,notes);
            throw new ServiceAuthOperationFault("Access Denied",null);
         }

        logAuthRequest("Success",username, ipAddr, serviceURI,notes);
   }
    catch (Exception ex){
                        throw new ServiceAuthOperationFault("Error serviceAuthOperation: " +
ex.getLocalizedMessage(),null);
    }
    return "ok";
    }
```

Figure A.13 (Part 2): Security Assurance Java Code

Appendix E: Replication Service, Database Binding, and BPEL

The replication service is implemented using composite service and BPEL, in which the source database access is done using one web service, and the update to the target database is done using another web service. Both web services use database binding component that comes with JBI in the OpenESB.

- WSDL file for source table service in the replication composite service:

Figure A.14 shows the WSDL file for service implementation of the source database which is the *EmpContacts*. The service performs the read of table rows using JBI database binding component over jdbc.

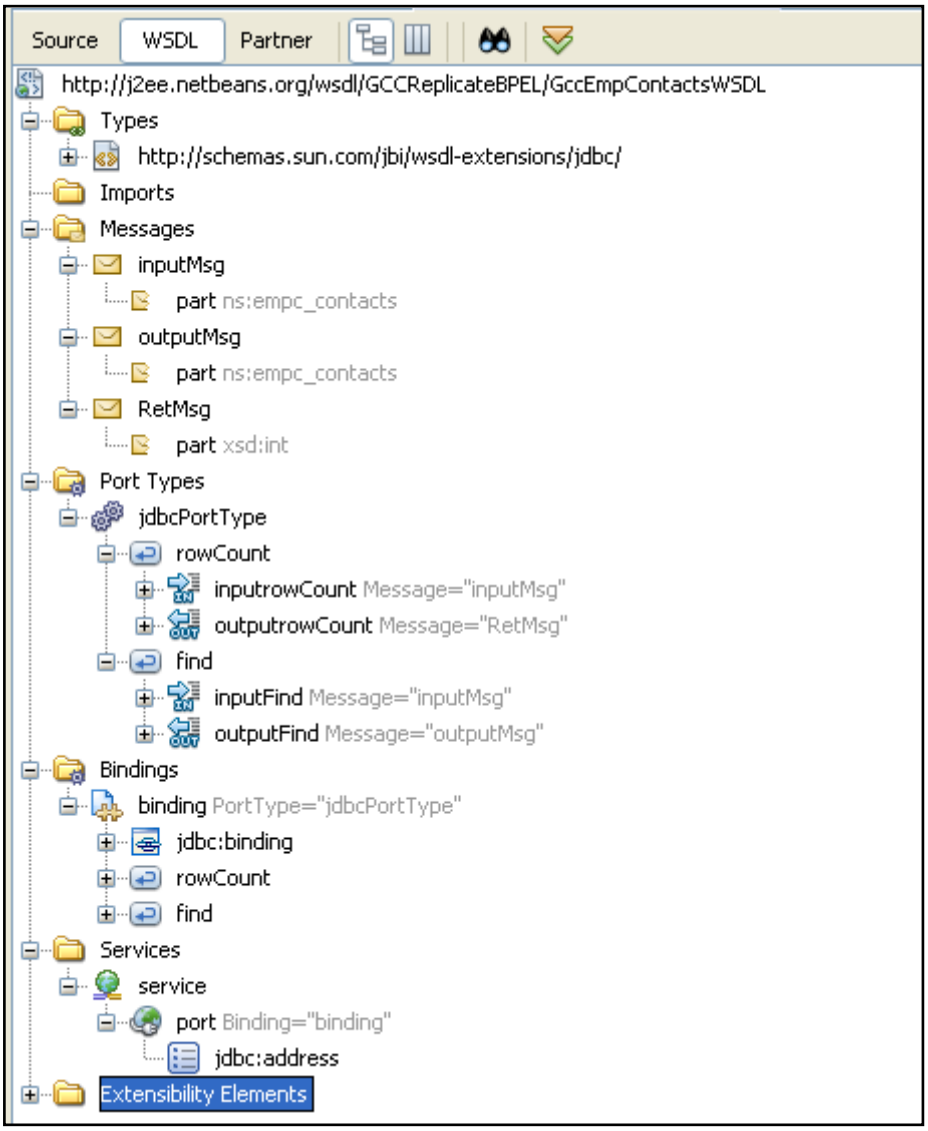

Figure A.14: WSDL File Source Table in the Replication Service

- WSDL file for target table service in the replication composite service:

This service performs the snapshot replication through deleting the content of the table in the database and inserting the records that are retrieved from the source table. This service performs delete and insert of the records in the table using the database binding component over JDBC binding. Figure A.15 depicts the WSDL of the service for the target table.

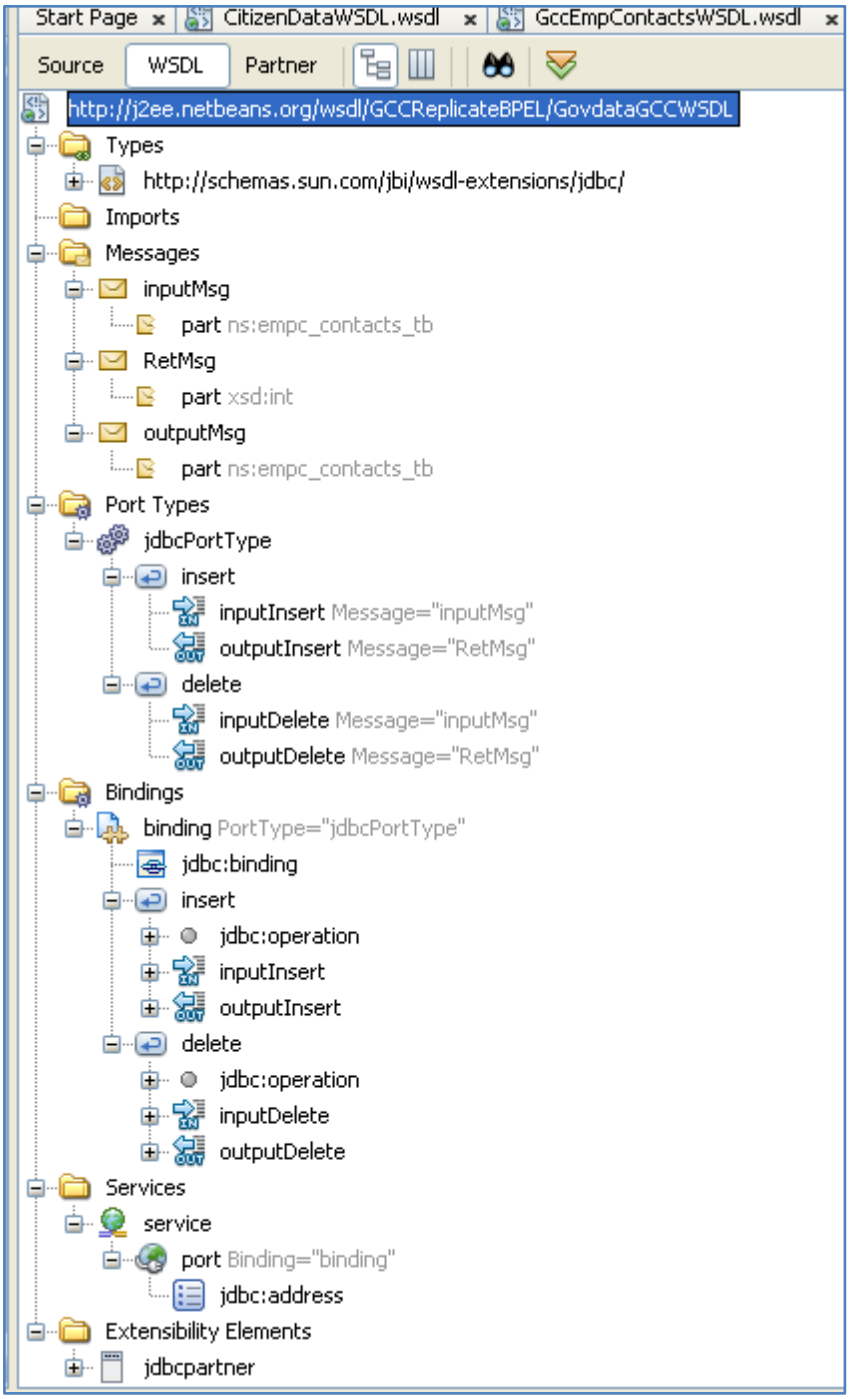

Figure A.15: WSDL File Target Table in the Replication Service

- Replication Service Composite Service BPEL Design View:

Figure A.16 shows the BPEL design for the composite web service that performs the replication between the source and target table. To the right of the Figure A.16 both target and source service with their operations. The operations in the source table service are *row-count* and *find*, and the operations in the target table are *delete* and insert. The replication composite service is shown in the left of the Figure A.16.

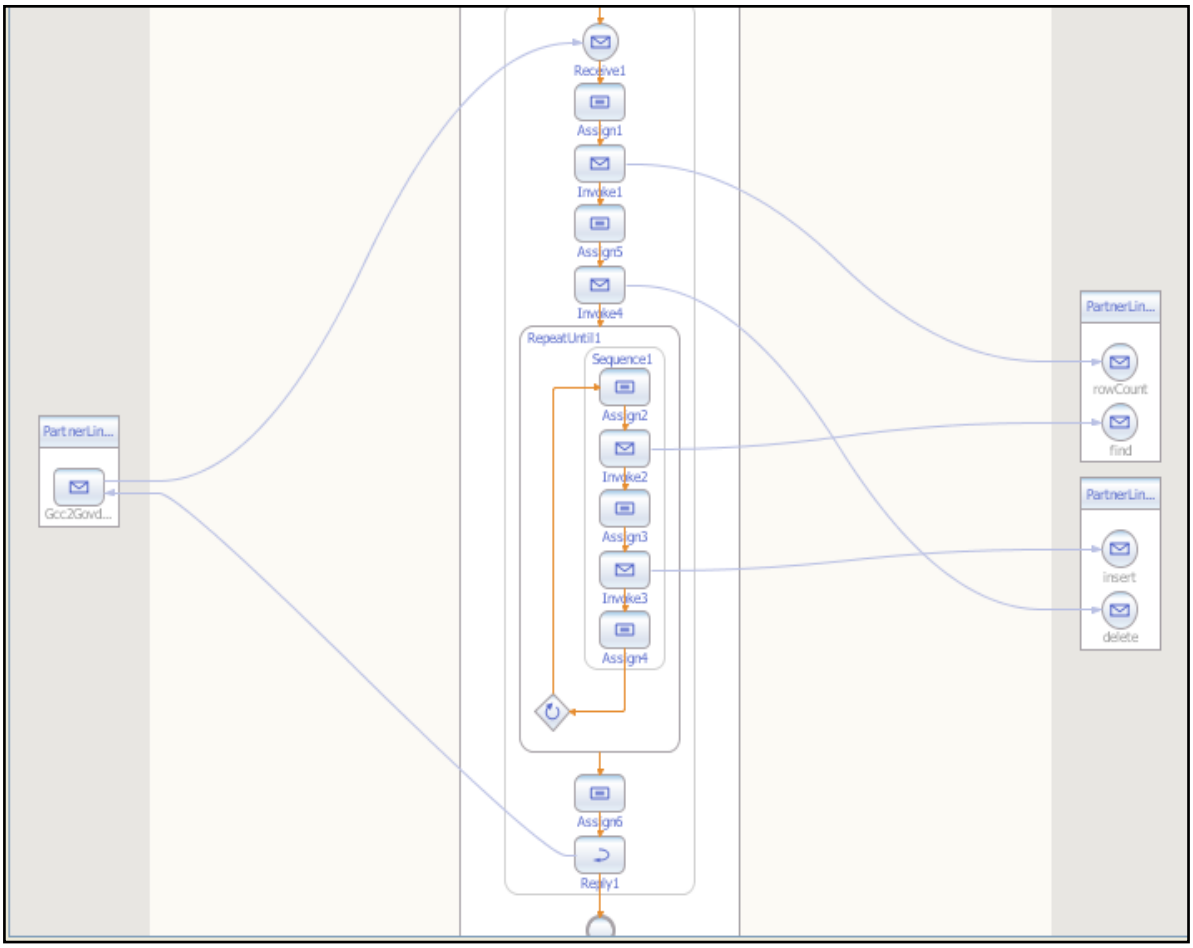

Figure A.16: BPEL for the Replication Service

Appendix F: Front-End Access Interface

- The Front-End interface is written in JSP and runs in the context of the web application service GlassFish. The interface allows the user to input an ID and returns the result for that ID in the Information Services: CitizenData, EmpData, and Insurance Data. Figures A.17, A.18, A.19 present snapshots from the front end interface.

- In Figure A.17 no connection to the GovDb exists and Error message is presented.

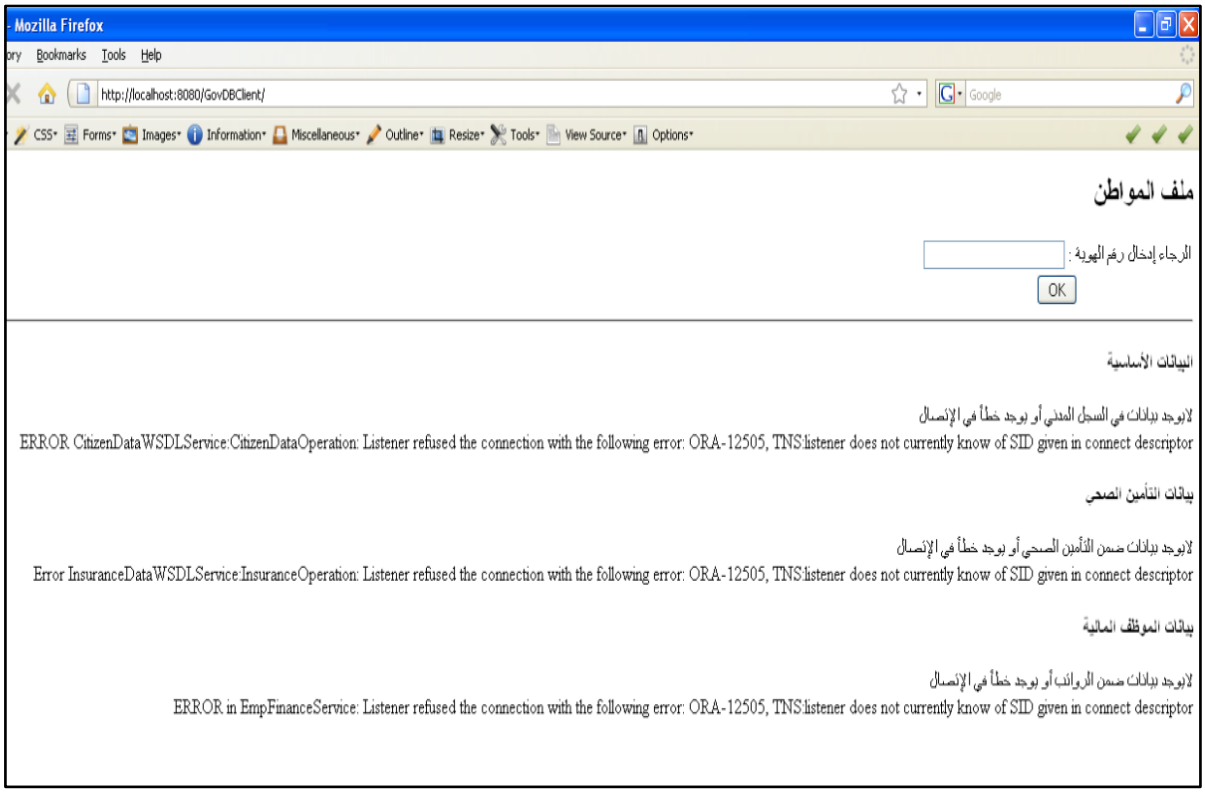

Figure A.17: Front-End Interface Error when *No-Database-Access*

- In Figure A.18 Invalid ID is searched for and Error Message Response is displayed

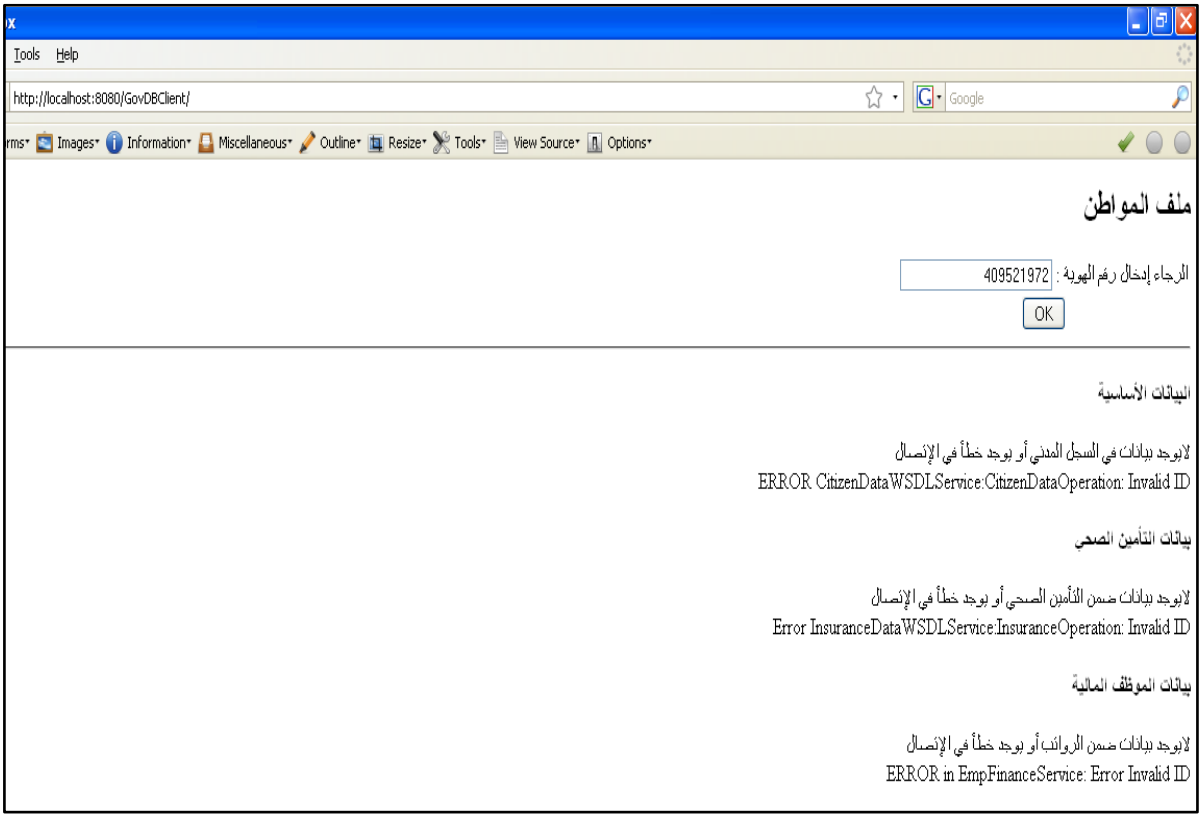

Figure A.18: Front-End Interface *Invalid-ID* Error Message

- Figure A.19 depicts a successful record retrieval from the three Informational Services.

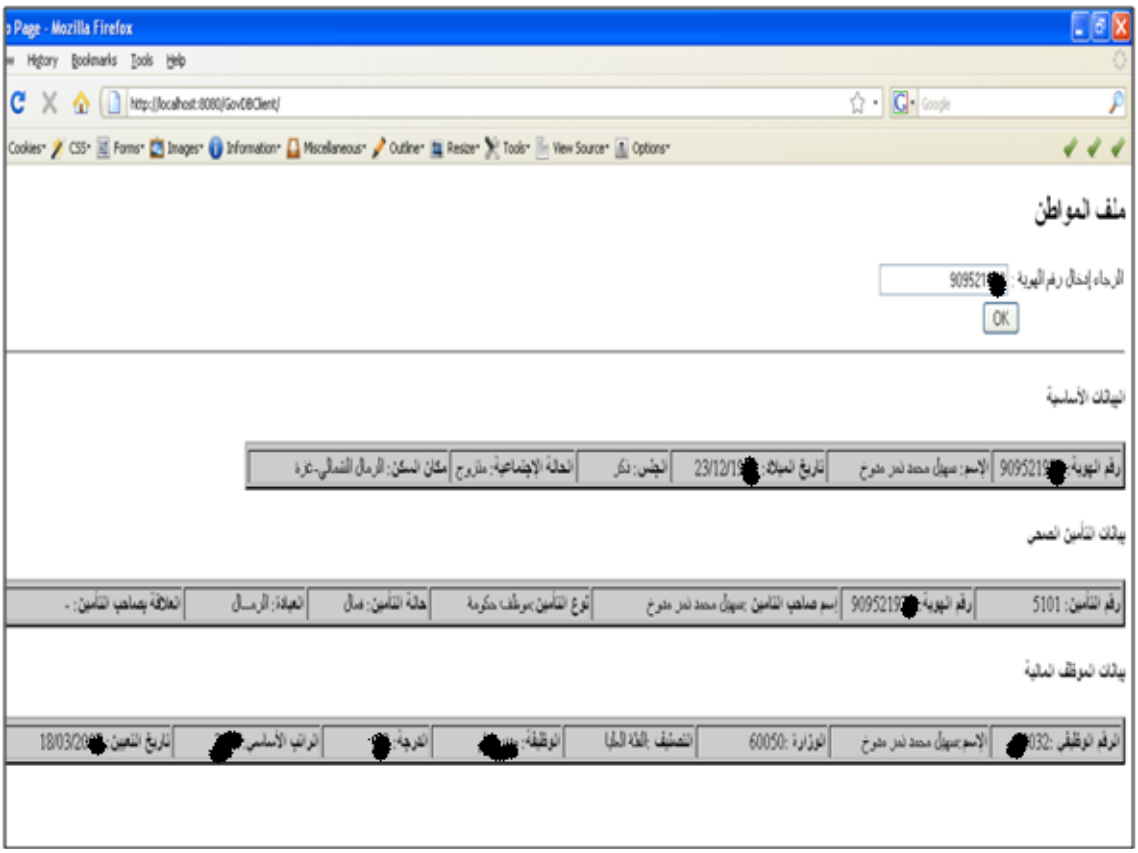

Figure A.19: Successful Record Retrieval from the *GovDb*

- Figure A.20 depicts the JSP Code for the front end interface to access the Informational Services

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Citizen Info Page</title>
    </head>
    <body dir="rtl">

        <h2>ملف المواطن</h2>
         <form name="form2" method="post" action="">
           <table border="0">
           <tr align="right"><td><%out.println(" الرجاء إدخال رقم
الهوية");%>:</td><td><input name="id" type="text" /></td></tr>
           <tr><td colspan="2" align="center"><input type="submit" value="OK"
/></td></tr>
           </table>
      </form>
             <%-- start web service invocation --%><hr/>

<%
     String  ino = request.getParameter("id");
      if ( ino != null ) {
        int i=0;
        try {
              i = Integer.parseInt(ino);
          org.netbeans.j2ee.wsdl.govdata.citizendatawsdl.CitizenDataWSDLService
service = new
org.netbeans.j2ee.wsdl.govdata.citizendatawsdl.CitizenDataWSDLService();
         org.netbeans.j2ee.wsdl.govdata.citizendatawsdl.CitizenDataWSDLPortType
port =     service.getCitizenDataWSDLPort();
        try {
        org.netbeans.xml.schema.citizendataschema.CitizenDataInType part1=   new
org.netbeans.xml.schema.citizendataschema.CitizenDataInType();
        part1.setId(i);
        part1.setUsername("user1");
        part1.setPassword("password1");
        org.netbeans.xml.schema.citizendataschema.CitizenDataOutType result1=
port.citizenDataWSDLOperation(part1);
        out.print("<h4>البيانات الأساسية</h4>");
        out.print("<table bgcolor=#CCCCCC dir=rtl  border=2 width=1000 > <tr> ");
        out.print("<tr align='right'>");
        out.print("<td width=200><b>رقم الهوية: </b>");
        out.print(result1.getCitizenID());
        out.print("</td>");
        out.print("<td width=300><b>الإسم: </b>");
        out.print(result1.getFirstName());
        out.print(" "+result1.getSecondName());
        out.print(" "+result1.getThirdName());
        out.print(" "+result1.getLastName());
        out.print("</td>");
        out.print("<td width=250><b>تاريخ الميلاد: </b>");
        out.print(result1.getBirthDate());
        out.print("</td>");
        out.print("<td width=150><b>الجنس: </b>");
        out.print(result1.getSGender());
        out.print("</td>");
        out.print("<td width=200><b>الحالة الإجتماعية: </b>");
        out.print(result1.getSMaritalStatus());
        out.print("</td>");
        out.print("<td width=350><b>مكان السكن: </b>");
        out.print(result1.getSRegion());
         out.print("-" + result1.getSCity());
```

Figure A.20 (Part 1): Front-End Application JSP Code

```
        out.print("</td>");
       out.print("</tr>");
        out.print("</table>");
        }
          catch ( Exception ex)
           {
             //out.println("<br>ERROR CitizenData: " + ex.getLocalizedMessage());
             out.print("<h4>البيانات الأساسية</h4>");
             String ErrorMsgMoi = " لايوجد بيانات في السجل المدني أو يوجد خطأ في
الإتصال";
             out.println(ErrorMsgMoi+"<br>"+ex.getLocalizedMessage());
           }
    try {
       org.netbeans.j2ee.wsdl.govdata.insurancedatawsdl.InsuranceDataWSDLService
service1 = new
org.netbeans.j2ee.wsdl.govdata.insurancedatawsdl.InsuranceDataWSDLService();
       org.netbeans.j2ee.wsdl.govdata.insurancedatawsdl.InsuranceDataWSDLPortType
port1 = service1.getInsuranceDataWSDLPort();
        // TODO initialize WS operation arguments here
       org.netbeans.xml.schema.insurancedataschema.InsuranceDataInType part2 = new
org.netbeans.xml.schema.insurancedataschema.InsuranceDataInType();
            part2.setId(i);
        part2.setUsername("user1");
        part2.setPassword("password1");
       org.netbeans.xml.schema.insurancedataschema.InsuranceDataOutType result2 =
port1.insuranceDataWSDLOperation(part2);
        out.print("<h4>بيانات التأمين الصحي</h4>");
        out.print("<table bgcolor=#CCCCCC  dir=rtl  border=2 width=1300 > <tr> ");
        out.print("<tr align='right'>");
        out.print("<td width=250><b>رقم التأمين: </b>");
        out.print(result2.getInsuranceID());
        out.print("</td>");
        out.print("<td width=200><b>رقم الهوية: </b>");
        out.print(result2.getCitizenID());
        out.print("</td>");
        out.print("<td width=450><b>إسم صاحب التامين :</b>");
        out.print(result2.getFName());
        out.print(" "+result2.getSName());
        out.print(" "+result2.getGName());
        out.print(" "+result2.getLastName());
        out.print("</td>");
        out.print("<td width=250><b>نوع التأمين:</b>");
        out.print(result2.getInsuranceType());
        out.print("</td>");
        out.print("<td width=200><b>حالة التأمين: </b>");
        out.print(result2.getInsuranceStatusDesc());
        out.print("</td>");
        out.print("<td width=200><b>العيادة: </b>");
        out.print(result2.getClinic());
        out.print("</td>");
        out.print("<td width=350><b>العلاقة بصاحب التأمين: </b>");
        out.print(result2.getRelTypeDesc());
        out.print("</td>");
       out.print("</tr>");
        out.print("</table>");
   }
          catch ( Exception ex)
           {
          //   out.println("<br>ERROR Insurance: " + ex.getLocalizedMessage());
              out.print("<h4>بيانات التأمين الصحي</h4>");
              String ErrorMsg="لايوجد بيانات ضمن التأمين الصحي أو يوجد خطأ في الإتصال
";
              out.println(ErrorMsg+"<br>"+ex.getLocalizedMessage());
           }
```

Figure A.20 (Part 2): Front-End Application JSP Code

```
try {
       org.netbeans.j2ee.wsdl.govdata.empdatawsdl.EmpDataWSDLService service2 = new
org.netbeans.j2ee.wsdl.govdata.empdatawsdl.EmpDataWSDLService();
       org.netbeans.j2ee.wsdl.govdata.empdatawsdl.EmpDataWSDLPortType port2 =
service2.getEmpDataWSDLPort();
        // TODO initialize WS operation arguments here
       org.netbeans.xml.schema.empdataschema.EmpFinDataInType part1 = new
org.netbeans.xml.schema.empdataschema.EmpFinDataInType();
        part1.setId(i);
        part1.setUsername("user1");
        part1.setPassword("password1");
       // TODO process result here
       org.netbeans.xml.schema.empdataschema.EmpFinDataOutType result3 =
port2.empFinDataWSDLOperation(part1);
         out.print("<h4>بيانات الموظف المالية</h4>");
        out.print("<table bgcolor=#CCCCCC dir=rtl  border=2 width=1300 > <tr> ");
        out.print("<tr align='right'>");
        out.print("<td width=250><b> الرقم الوظيفي :</b>");
        out.print(result3.getEmpNo());
        out.print("</td>");
        out.print("<td width=300><b>الإسم:</b>");
        out.print(result3.getEmpName());
        out.print("</td>");
        out.print("<td width=250><b>الوزارة :</b>");
         out.print(result3.getMinistry());
        out.print("</td>");

        out.print("<td width=250><b>التصنيف :</b>");
         out.print(result3.getScale());
        out.print("</td>");
        out.print("<td width=250><b>الوظيفة: </b>");
         out.print(result3.getDept());
        out.print("</td>");
        out.print("<td width=200><b>الدرجة: </b>");
          out.print(result3.getGrade());
        out.print("</td>");
        out.print("<td width=250><b>الراتب الأساسي:</b>");
       out.print(result3.getSal());
        out.print("</td>");
        out.print("<td width=350><b>تاريخ التعيين: </b>");
        out.print(result3.getHireDate());
        out.print("</td>");
       out.print("</tr>");
        out.print("</table>");
    } catch (Exception ex) {
         //out.println("<br>ERROR EmpFin: " + ex.getLocalizedMessage());
          out.print("<h4>بيانات الموظف المالية</h4>");
          String ErrorMsgMof = "لايوجد بيانات ضمن الرواتب أو يوجد خطأ في الإتصال";
          out.println(ErrorMsgMof+"<br>"+ex.getLocalizedMessage());
       // TODO handle custom exceptions here
    }
 } catch ( Exception ex){
              out.println("ERROR Input: " + ex.getLocalizedMessage());
          }
        }
    %>
    </body>
</html>
```

Figure A.20 (Part 3): Front-End Application JSP Code

Appendix G: Framework Evaluation Scenarios based on ATAM

I. Interoperability Evaluation Scenarios

The evaluation of Interoperability for the SOA framework should consider the following concerns and their prompts:

1. General Scenario 1:

   Question: Does the framework provide support to use services implemented in disparate platforms and using different languages?

   Prompt: The framework is designed to support diverse services implemented in various platforms and languages. Web Services provide primarily syntactic Interoperability. Whether two components can interoperate also depends on their semantic agreement about the meaning of data and operations. Web Services technology use SOAP and WSDL standards that can be used between different service providers and users regardless of the need or be aware of the underlying development language.

2. General Scenario 2:

   Question: Does the framework have the ability to replicate and access databases with heterogeneous database type?

   Prompt: The framework allows replicating heterogeneous database types, whether using Oracle, MS-Sql, MySQL, and so on. It is designed to use general database connectivity interface which can access any database that support JDBC, and depends on the list of data sources supported by the connectivity driver.

3. General Scenario 3:

   Question: Does the framework have the ability to orchestrate services implemented in disparate platforms and using different languages?

   Prompt: The framework is using BPEL for business process which can orchestrate web services that uses SOAP and WSDL for service interfacing regardless of the underlying platform or development languages. The BPEL engine (Orchestration Engine) allows systems with disparate underlying platforms (e.g., Java and .NET) to interact through Web services technology.

4. General Scenario 4:

   Question: Does the framework allow having service users and providers to use different implementation languages and platforms?

   Prompt: Both service users and providers are unaware of their counter platform and development languages, the only requirement between them is to have a unified standard that for providing and invoking the service. This is achieved via WSDL, SOAP and HTTP. Moreover, in the framework, we are using SOAP document-literal which is more interoperable than RPC-encoding due to incompatibility in SOAP encoding across platforms.

5. General Scenario 5:

   Question: Will the framework middleware (integrator) allows connected applications with disparate technology and data formatting requirements to interoperate as service users and providers without major changes to each?

   Prompt: The middleware integration approach in the proposed SOA based framework will be an ESB. It is hub-and-spoke SOA approach, which would allow connecting diverse applications, technologies, and data formatting.

6. General Scenario 6:

   Question: Does the framework allow for compatibility between authentication mechanisms supported by participant of the service providers?

   Prompt: Authentication mechanism will be centralized and realized using Web Service; the authentication and access information is managed centrally. The authentication access is interoperable since it is based on Web Services approach and not other general purpose authentication schemas such as LDAP or Active Directory, and so on.

7. General Scenario 7:

   Question: Does the framework use any standards to support message-level security (e.g., WS-Security, XML Encryption, and XML Signature)?

   Prompt: The framework is designed to support such message-level security; it is left for the service provides to implement such features to further enhance the security of the service.

8. General Scenario 8:

   Question: Is the interaction between a given service user and provider synchronous or asynchronous?

   Prompt: The framework provides support for both synchronous and asynchronous web service. It is left for the requirement and operation of the service to use either of them. The services implemented in the prototype are synchronous services.

9. General Scenario 9:

   Question: Do legacy systems integrate with the framework?

   Prompt: The ESB, the middleware integrator, is responsible for integrating legacy systems to the framework, which provides Interoperability with old legacy applications.

10. General Scenario 10:

    Question: Which standards provide Interoperability is used in the framework?

    Prompt: The standards used in the framework and provide Interoperability between framework components when interacting with each others are: WSDL, SOAP, UDDI, and BPEL which provide capabilities to systems developed with Web services technology.

11. General Scenario 11:

    Question: What challenges Interoperability in the framework?

    Prompt: Not all Web services platforms implement the same version of the additional standards such as UDDI, BPEL, WS-Security and hence achieving Interoperability faces some obstacles when using such standards. Still since the framework is under a centralized unit of administration, this risk can be mitigated.

II. Manageability Evaluation Scenarios

The evaluation of Manageability quality attributes for the proposed SOA based framework should consider the following concerns and their prompts:

1. General Scenario 1:

   Question: Does the framework support central point of service access management?

   Prompt: The ESB supports a centralized point of management of the services. The System Management Service also provides management capability for metric usage and health monitor of the framework services.

2. General Scenario 2:

   Question: Does the framework provide logging access capability?

   Prompt: The Management Service in the framework provide the capability of accessing all logs related with the services usage, and provides presentation logic for the framework logs repository.

3. General Scenario 3:

   Question: Does the framework allow for central authentication directory?

   Prompt: The framework allows for an authentication that is centralized through using the Security Assurance Service, the authentication can be used by all services, and hence provide a central point of authentication.

4. General Scenario 4:

   Question: Does the framework provide metric usage?

   Prompt: The framework provides metric usage when using Security Assurance Service which records a log of the services access, in addition to this the ESB and the application server provides a metric usage and logging of services usage.

5. General Scenario 5:

   Question: Does the BPEL process and environment provide support for monitoring and logging event data to allow the measurement of business metrics?

   Prompt: The BPEL engine manages BPEL processes and the application server that runs the engine in its context provides monitoring and logging of event data and measurement of business metrics such as wait time, transaction volumes, and exception counts.

6. General Scenario 6:

   Question: Does the framework support monitoring facility?

   Prompt: The framework through the ESB and application server provides a monitoring facility for the invoked services, and the health of the framework components.

7. General Scenario 7:

   Question: Does the framework support stop/start components and services?

   Prompt: The ESB and the application server under which the services run allow to start and stop framework engines, components, and services.

8. General Scenario 8:

   Question: What support exists for monitoring and event data logging?

   Prompt: The mechanisms for monitoring and event logging allow taking measures, such as wait times, transaction volumes, and exception counts. These measures are important to oversee the system in production and for the testing of reliability and performance analysis.

III. Flexibility Evaluation Scenarios

The evaluation of Flexibility quality attributes for the proposed SOA based framework should consider the following concerns and their prompts

1. General Scenario 1:

   Question: Is the code of framework components easy to change, when a desired change has been determined?

   Prompt: Since the framework is composed of diverse components, most of them are Web Services, which are self-contained and loosely coupled; the changes required for any service would be incur little efforts.

2. General Scenario 2:

   Question: Are BPEL processes designed with proper decoupling between services?

   Prompt: Services in the process can be changed without affecting every other service in the BPEL workflow, as far as the input/output types of the service are not changed, this enhances Flexibility.

3. General Scenario 3:

   Question: How is the identity and access information used by service implementations, such as IDs and passwords, stored?

   Prompt: The identity information is not hard-coded in services implementation. Depending on the realization of the Web Service, access information can be stored in an XML file or database data source.

4. General Scenario 4:

   Question: Do the framework provide loose coupling between services?

   Prompt: The services to be implemented are coarse grain and hence self contained and can operate independently, hence provides loose coupling and enhances Flexibility.

5. General Scenario 5:

   Question: Is it easy to manage and present credentials such as passwords, certificates, and tokens if used in the framework?

   Prompt: The credentials are used in Security Assurance Service; they are easily managed and not hard coded. They are stored in an XML format, which is flexible for manipulating and easy for understanding. It is centrally configured by an administrator, and can be managed through the System Management Service

6. General Scenario 6:

   Question: Is it possible to access the framework from diverse networks or non-governmental network?

   Prompt: Access to services by consumers can be from the Internet as an open and insecure network, as well as, from the private governmental network. Security restriction is imposed by Security Assurance Service to guarantee security of the information. This increases the Flexibility of the framework, as it can be used over diverse networks.

7. General Scenario 7:

   Question: Is it possible to use diverse data sources in the framework?

   Prompt: Using a new data source in the framework, or adding another database type can be achieved with little efforts, because a minimal change is required in the code that access the database, if such database connectivity is not supported by the JDBC.