UNIVERSITY OF NAIROBI
COLLEGE OF BIOLOGICAL & PHYSICAL SCIENCES

SCHOOL OF COMPUTING AND INFORMATICS

AN ELECTRONIC GOVERNMENT INTEGRATED SYSTEM USING SERVICE ORIENTED ARCHITECTURE (SOA)

STUDENT NAME: VIOLET L. MASAVA

STUDENT NUMBER: P58/61718/2010

SUPERVISOR:
PROF. W. OKELLO-ODONGO

JULY, 2013

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTERS OF SCIENCE (MSC) IN COMPUTER SCIENCE OF THE UNIVERSITY OF NAIROBI

DECLARATION

STUDENT

I, the undersigned, declare that this research project is my original work and that it has NOT been presented in any other university or institution for academic credit.

Signature …… Date ……...
VIOLET L. MASAVA
P58/61718/2010

SUPERVISOR
This research project has been submitted for examination with my approval as the University supervisor.

Signed …… Date ……
PROF. W. OKELLO-ODONGO

ABSTRACT

The aim of the study was to investigate the application of Service Oriented Architecture (SOA) as an integration solution to disparate departmental government information systems with the aim of improving service delivery. The specific objectives that the study addressed include identification of business work flows, communication flows, and the common information requirements within the three departments of the Ministry of Immigration and Registrar of Persons; IMD, CRD and NRB; identification of the current and potential integration difficulties of the three systems; development of a SOA model of the proposed integration solution; building a system prototype using JAVA SOAP web services to implement the model; testing, evaluation and validation of the prototype. The data collection involved interviewing staff at the ministry department and carefully documenting all the processes. The findings of the study revealed that service delivery can be significantly improved by the adoption of Service Oriented technology. These findings have multiple implications on service delivery and also lowering of costs related to system development.

ACKNOWLEGDEMENT

Firstly, I would like to thank the Almighty God for His unending grace, strength and patience as I undertook this study. I would not have come this far without Him.

Many thanks also extend to my family for their immense moral and financial support through the entire project period.

My heartfelt gratitude to my project supervisor Prof. W. Okelo-Odongo for his sacrificial guidance and Deputy Director, School of Computing and Informatics, Mr. C.A. Moturi for the invaluable time in polishing my work.

The input of dedicated colleagues, friends and the officials at MIRP into the completion of this study cannot go unnoticed.

I’m greatly indebted to you all for every effort towards the completion of this project.

May God richly bless you all.

ACRONYMS

CORBA	-	Common Object Request Broker Architecture
CRD	-	Civil Registration Department
DCOM	-	Distributed Component Object Model (DCOM)
GUI	-	Graphical User Interface
HTTP	-	Hyper Text Transfer Protocol
IMD	-	Immigration Department
MIRP	-	Ministry of State for Immigration and Registrar of Persons
NRB	-	National Registration Bureau
SOA	-	Service Oriented Architecture
SOMA	-	Service Oriented Modeling Architecture

DEFINITION OF TERMS

1. Service

A service is a discrete unit of business functionality that is made available through a service contract.

2. Web Services

Web services are the amalgamation of eXtensible Markup Language (XML) and HyperText Transfer Protocol HTTP that can convert an application into a Web-application, which publish its function or message to the rest of the world.

3. Extensible Mark Up Language (MXL)

It is an independent standard data format for describing data to be exchanged on the web. As its name indicates, it is a markup language that involves the use of tags that "mark up" the contents of a document and in doing so describes the contents of the document.

4. Web Services Description Language (WSDL)

The WSDL refers to Web Services Description Language, is an XML based protocol used for sending and receiving the information through decentralized and distributed environments. It defines what services are available in its Web service and also defines the methods, parameter names, parameter data types, and return data types for the Web service. The WSDL document is quite reliable and applications that use web services accept it

5. Simple Object Access Protocol (SOA)

It is an XML - based standard interoperability protocol that is used for exchanging of information in a distributed environment.

It provides a common message format for exchanging data between clients and services.

6. Universal Description Discovery Integration (UDDI)

It is a platform-independent, Extensible Mark up Language (XML)-based registry for businesses worldwide to list themselves on the Internet and a mechanism to register and locate web service applications.

7. Remoter Procedure Call (RPC)

Remote procedure call 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.

FIGURES

TABLES

CHARTS

GRAPHS

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND

Over the last 10 - 15 years, most organizations and government establishments in Kenya have moved from using information systems which were predominantly built and maintained in-house to the purchase of products from external vendors. Integrating these systems is therefore an increasingly important agenda for most of these organizations and government establishments. Systems Integration ensures that operations and businesses can begin to operate more holistically, with systems working together to build cohesive and deeper relationships between service provision entities and users for greater co-operation, speedier access to data and a much better grasp of resources.

The demand for information systems integration therefore stems from a number of sources:

- The need of organizational management for coherent management information that has been lacking and consequently lowering the quality of key decision making;
- Increasing expectations that systems can 'seamlessly' support 'the user experience' who are the foremost recipients of the systems and;
- Efforts to eradicate duplication or expensive re-keying of data which has currently caused a surging increase in overheads.
- Once systems integration is realized in organizations and across government agencies, the following numerous benefits are reaped that go into addressing the aforementioned disintegration issues;
- Less concern with being locked in to a single vendor who can be manipulative for selfish commercial gains.
- Seamless flow of information between systems and their respective users
- Reduce duplication of processes and documents pointing towards reduced overheads
- Greater co-operation between departments
- Ability to add greater value to each transaction

Systems integration as an integral part of E-government

The Kenyan government has adopted numerous ICT enabled reforms in order to deliver greater and improved public access to government's services and information while making the government more accountable to its subjects. *(Kenya Vision 2030 secretariat, 2006).*This has led to the rapid rise of information technology and its adoption into the government sector that has necessitated various government bodies, departments and functional administrative units to acquire numerous heterogeneous and usually technically incompatible IT systems that not only need to work together within the government, but can also be accessed from outside by the citizens and other relevant entities.

Systems integration is therefore an integral part of E-government that ensures that the above deliverables are achieved and glued together. It is unfortunate to note that systems integration has however had a slow rate of diffusion and acceptance in most developing countries, Kenya included. This is generally due to;

- Users and practitioners skepticism on its actual benefits.
- Resource issues, including the costs of internal staff and outsourcing of external support services;
- Lack of adequate necessary in-house skills and expertise that are needed for systems integration (particularly acute in the IT sector);
- Internal resistance from staff and service departments determined to 'do their own thing' and protect 'their' data for self-seeking motives;
- Lack of an aggressive representation of the integration issue at a senior management level; and, a lack of appreciation in parts of the organization of the multiple uses to which data is put and therefore a tendency to be concerned with the adequacy of data for local purposes only.

1.2 PROBLEM STATEMENT

The government of Kenya currently faces the challenge to manage a growing portfolio of information systems across its operations spectrum. The heterogeneity of these systems manifest at various levels: different data formats and structures- flat files, database schemas, XML, different platforms (operating systems, compiler and middleware) often tied to the vendor of the system; some open source while others closed. The costs of acquiring, interoperating and maintaining these systems are rising, while the demands of system users are exponentially increasing.

The burden of integration usually falls on the users of the system, who are forced more often than not to access many different sub systems to accomplish a single task. Many shortcomings arise as a result of this disintegration;

- Hampering delivery of services to the intended parties in an efficient, reliable and timely manner.
- Inaccurate, untimely and inconsistent management information that would otherwise be a key component in decision making at various hierarchies o governmental entities.
- Increased inefficiency in duplication of effort and resources.
- Dreadful end user (staff) experiences who are the ones that bear the most burden of iterating tasks.

This therefore calls for the undisputable need for integration of various governments' IT systems in order to deliver essential government services to citizens and other intended parties in an efficient and effective manner.

Ministry of State for Immigration and Registration of Persons (MIRP) like many other government entities is a key ministry within the government that has for years grappled with information systems disintegration issues. It is primarily responsible for overseeing the operations of its three major departments; Immigration department (IMD), National Registration Bureau (NRB) and the Civil Registration Department (CRD).

These three departments currently use different information systems which have led to islands of information with little or no information sharing and as a result, communication between the different systems is nonexistent and yet they are unequivocally interdependent. For instance, a citizen's information at the CRD; birth certificate details are the key identifying information needed by the NRB to process the same citizen's national identification card. The IMD heavily depends on these two other departments for the citizen's information in order to process a passport for the same citizen.

It is therefore imperative for these three systems to be integrated in order to improve service delivery to the citizens and also provide a friendlier environment for their respective staff to work.

This project considered consider Service Oriented Architecture (SOA), as an architectural design pattern that to address the integration issues across the three departments in the Ministry of State Immigration and Registrar of Persons (MIRP), in order to improve the delivery of services to citizens other intended parties and also data sharing between the three different departmental units at MIRP with. SOA as an integration method stands out from the rest due to its unparalleled benefits that include; loose coupling that addresses the interoperability and platform independency, location transparency and reuse of services that other integration methods have failed to address.

1.3 OBJECTIVES

Overall Objective

To investigate the application of Service Oriented Architecture (SOA) as an integration solution to disparate departmental government information systems with the aim of improving service delivery.

Specific Objectives

- **a)** Identify the business work flows, communication flows, and the common information requirements within the three departments of IMD, CRD and NRB.
- **b)** Identify the current and potential integration difficulties of the three systems and services.
- **c)** Develop a SOA model of the proposed integration solution.
- **d)** Build a system prototype using JAVA SOAP web services to implement the model.

1.4 SCOPE OF PROJECT

Whereas MIRP has five departments, this project investigated the development of a prototype that offers an integration solution to three disparate systems in the immigration ministry using Service Oriented Architecture framework. The other two departments; Refugee Affairs Department and the Population Registration Services were not considered in this project since their core services are not primarily to the majority population of citizens.

1.5 JUSTIFICATION OF STUDY

The Ministry of State Immigration and Registration of Persons has invested lots of money in automating their service delivery systems to their citizens. However this has not reflected in improved service delivery. Citizens have to wait for days for confirmation of details from other departments. Thus the main challenge is the need for interoperability among different legacy systems that operate on different technologies and the need to offer services to citizens in a transparent yet efficient manner. By tackling the integration from a Service Orientation perspective, this study increased the service levels of the immigration ministry while being mindful of the cost, technical as well as social challenges involved.

The main expected contribution of this study in the integration and SOA community was:

a) A unified and transparent architecture for information sharing between the different departments in the ministry of State Immigration and Registration of Persons.
b) Transparent and efficient service delivery architecture for integrated services.
c) A practical system implementation of the SOA in integration using Java SOAP web services.

CHAPTER TWO

LITERATURE REVIEW

2.1 E-GOVERNANCE

E-Governance is the use of Information and Communication Technology (ICT) to enable more efficient, reliable, cost-effective means of a government service delivery to its citizens (G2C), business entities (G2B) and to other inter-governmental agencies (G2G*) (Jeong, 2007).*

The Kenya government has taken these ICT enabled reforms to allow for greater and improved public access to government's services and information while making the government more accountable to citizens.

The greatest benefit that is reaped from e-governance innovations and practices is the reinforcement of other reforms that would help a country to better compete in both the regional and global fronts in strengthening its social, economic and political pillars.

Each government is mandated to deliver numerous services in various sectors. These sectors include but are not limited to: Financial sector, Health sector, Agricultural sector, Education sector, Tourism sector, National Registration Bureau, Civil Registration Department, Immigration Services etc.

Systems integration is therefore the spring board from where the effective delivery of these government services can be realized and also the glue that will hold them together.

2.2 SYSTEM INTEGRATION

System integration is defined as putting diverse hardware and/or software components together to work as a system (O'Brien, 2008). Integration is a complex task, especially when the applications involved reside on older legacy systems or utilize different hardware or software platforms.

2.2.1 Background of Integration Technology

From a historical perspective, the earliest computer systems were large stand-alone computers known as mainframes that ran only one computer program at a time. Multiprocessing, the ability to run several programs, each in a distinct partition of the mainframe's memory, was a technical breakthrough that arrived in the 1960s. Since then, businesses have continued to require ever more computing power and flexibility, and the level of complexity of software solutions has increased significantly.

Each decade since the 1960s has seen advances in computer technology, with each generation of hardware and software solutions standing on the shoulders of prior developments. This constantly changing environment creates a continuing dilemma for businesses of all sizes in all areas. This article traces the evolution of integrated computer systems (Beach, 2004).

2.2.2 Integration Impact on Organizations.

Computer and information technology remains a major organizational expenditure in terms of initial investment and continuing maintenance costs *(Goodhue, 1992).* Integration bridges the gap between older legacy systems that continue to function and newer technologies that have been developed along the way.

Organization leaders must therefore continually evaluate the pros and cons of when to adopt the latest technologies.

Organizations use integration technology to pull together applications and extract greater benefits from their computer systems. Integration can result in cost savings, additional revenue, and competitive advantage when techniques such as process automation and business monitoring are utilized. Organizations may additionally reap benefits from rationalizing applications following an acquisition or merger, where each of the combined businesses has been using proprietary solutions, (O'Brien, 2008).

2.3 SYSTEM INTEGRATION LEVELS

There are four common system integration levels, (Linthicum, 2008):

- Data-level integration
- User interface (UI)-level integration
- Application-level integration
- Method-level integration

2.3.1 Data-Level Integration

The backend data stores of the relevant application are integrated, and can be either push or pull based. When using push based, one application makes SQL calls on another application's database tables. This is through database links or stored procedures, and data is pushed into another application's database. However, pull based integration uses triggers and polling. The triggers capture changes to data and write the identifying information to interface tables. It is then possible for adaptors to poll the application's interface tables and retrieve the pertinent data. This pull based integration is used when an application requires passive notification of changes within another application's data. When the application that needs to be integrated does not provide any APIs or client interfaces, you would use data-level integration. You must also have a good understanding of the business operations that may affect the application's data model.

It is typically the only option with most custom applications that lack APIs.

2.3.2 User Interface-Level Integration

This ties integration logic to user interface code, and can be either scripting or proxy based. When using scripting based, the integration code is embedded into the user interface component events, common with client/server applications such as PowerBuilder.

In cases where direct access to the database is not easy or possible, or when the business logic is embedded in the user interface, this is the correct integration method to use. Mainframe and client/server applications are often good candidates for this. Mainframes do not tend to have access to friendly data stores, and do not provide public APIs. However, user interface level integration is generally used as a last resort. If you add scripting logic to catch events with client/server applications they become very difficult to maintain, as integration levels increase and more changes occur. User interface changes can break integration triggers and logic anyway. This tight coupling creates a permanent link between the maintenance of the user interface and the integration code.

2.3.3 Application-Level Integration

This is considered the best way forward for application integration, and it uses the integrated application's integration frameworks and APIs. It is good to use, since it is transparent to the integrated application and it preserves the application's data integrity. The application interface allows you to invoke business logic to preserve data integrity.

2.3.4 Method-Level Integration

This is less frequently used specialization of the application level integration method shown above. Here, we aggregate common operations on multiple applications into a single application that fronts the integrated applications. It is generally used when each integrated application has a similar set of API or functional methods. The integrated applications must support a Remote Procedure Call (RPC) or distributed component technology. The main disadvantage to this approach is again the tight application coupling in front components. They will break when changes are made to the integrated application API, and these problems will propagate down to the other applications that rely on them. This is used when we have distributed component or CORBA technology.

2.4 SYSTEMS INTEGRATION ARCHITECTURES

In a well-designed building, the electrics and plumbing usually keep working no matter how many appliances are switched on. Such a building is also capable of extension without having to tear up the blueprints and start again because of its good architectural design.

The same applies to software systems. Software architecture is the backbone of any complex computer system. The architecture encompasses all of the software elements, the relationships between the elements and the user interfaces to those elements. The performance and reliability of a software system are highly dependent upon the software architecture (Clements, P., *et al*, 2001).

Well-designed software architecture can be extended with relative ease to accommodate new applications without requiring extensive infrastructure development.

Described herein are the most common **"Integration"** Software Architectures (O'Brien, 2008).

2.4.1 Point-To-Point Integration Architecture

The original architecture used to support systems integration was called Point-to-Point. It derives its name from the direct, tightly bound connections that are made between applications and is the simplest of the integration architectures.

Point-to-Point Architecture

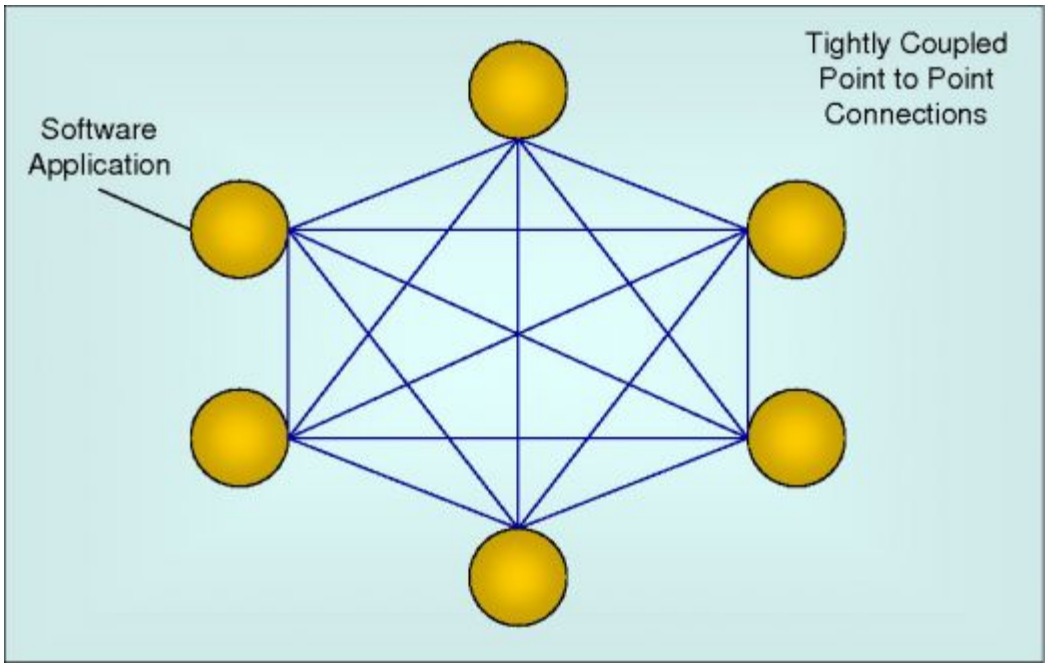


Fig. 2.1 Point-to-Point Integration

2.4.2 Hub and Spoke Integration Architecture

The earliest formal integration technologies worked on the principle that all information coming from the applications had to be processed within a single machine or server called a "hub". Acting as a central point of control, the hub dealt with all message processing including routing, splitting and combining of messages, mapping, and so on.

Hub and Spoke implementations decouple the sending and receiving applications. Unlike Point-to-Point, the applications on either side of the hub can be modified independently of each other. Since applications no longer need to perform data mapping, centralized definition and control of business processes could be easily achieved for the first time.

Hub and Spoke Architecture

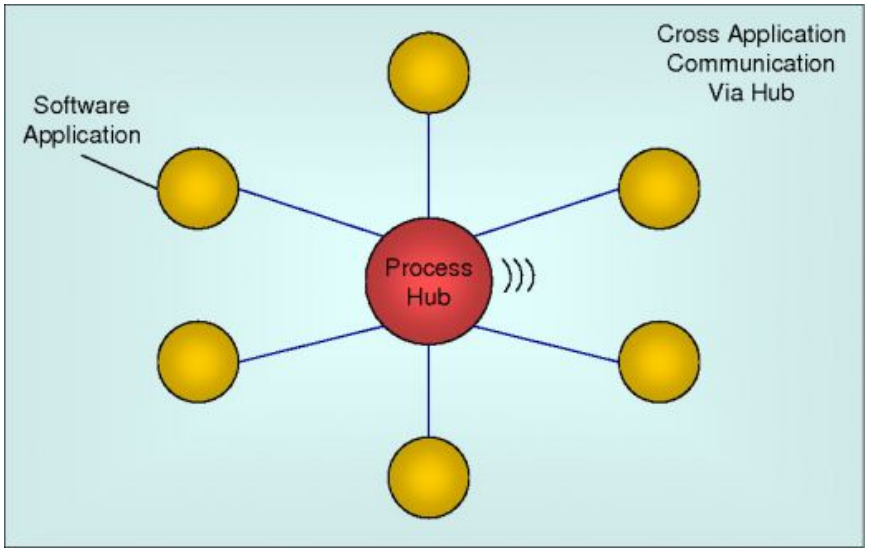


Fig. 2.2 Centralized Integration Processing Hub

2.4.3 Distributed Integration Architecture

One solution to the Hub and Spoke scalability issue is to perform message translation, routing, splitting, and combining closer to the source and target systems by using smaller computers known as "agents." Agent computers are connected to just one system and reduce the processing load on that system. Agents take information from the application they are connected to, process it, and send it to any target application(s) interested in receiving that information. The end result is Distributed Architecture. It is also known as **Peer to Peer** architecture.

Distributed Integration Architecture

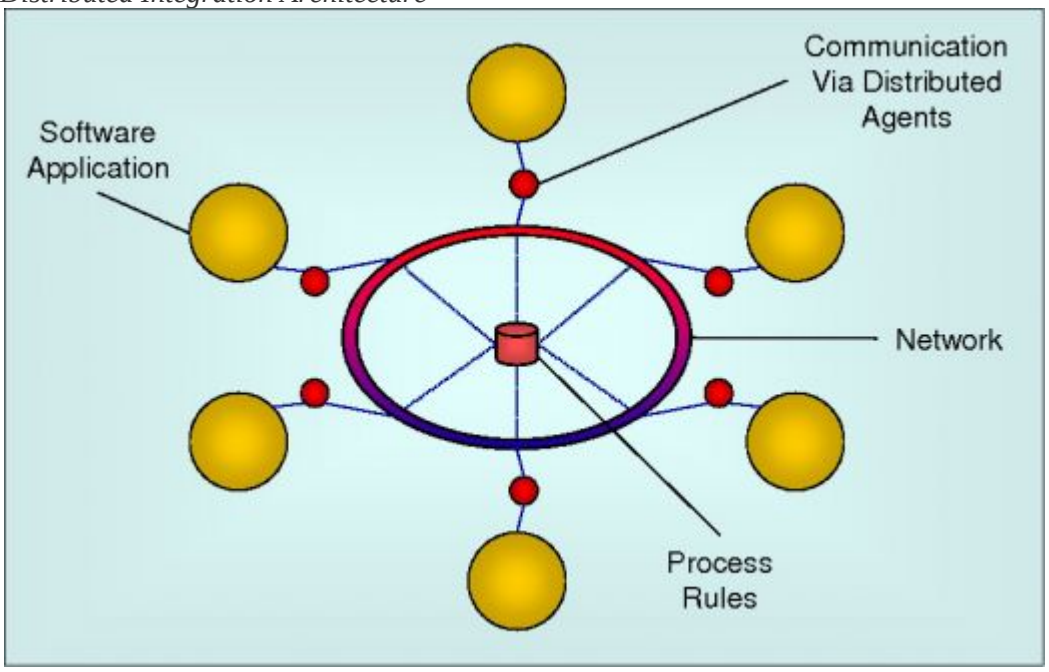


Fig. 2.3 Distributed Integration Processing Agents

Early attempts at Distributed Architecture would only work where the internal and external facilities operated under the same distributed technology. The choices available were:

1) **CORBA (Common Object Request Broker Architecture):** This was the Object Management Group's (OMG) open, vendor-independent architecture and infrastructure that computer applications employed to work together over networks.
2) **Microsoft's COM (Common Object Model):** This was later to become Distributed COM (DCOM) and a transactional version called COM+ was created later still. All of these have subsequently been combined with Microsoft's .NET initiative.
3) **Java RMI (Remote Method Invocation)**Java RMI enabled the creation of distributed Java-based to Java-based applications. It is now available in several software packages from Sun Systems.

2.4.4 Service Oriented Architecture (SOA)

Service Oriented Architecture (SOA) is the latest architectural approach, although it's not really very new. Service Oriented Architecture is essentially an enhanced version of Distributed Architecture that uses loosely coupled software services to support the requirements of business processes and software users. It goes a

step further than the previous architectures by providing an integrated environment which spreads out the workload, breaking down the different "silos" of business functionality and opening their processes to other applications.

One way to think of SOA is like a Lego set. A Lego set is more than just the individual blocks; specialized bigger pieces for complex creations are also available. Similarly, with SOA, an application can customize and/or change the individual pieces or "services" that it uses. Using these concepts, vast and complex component based applications can be developed.

Service Oriented Architecture (SOA)

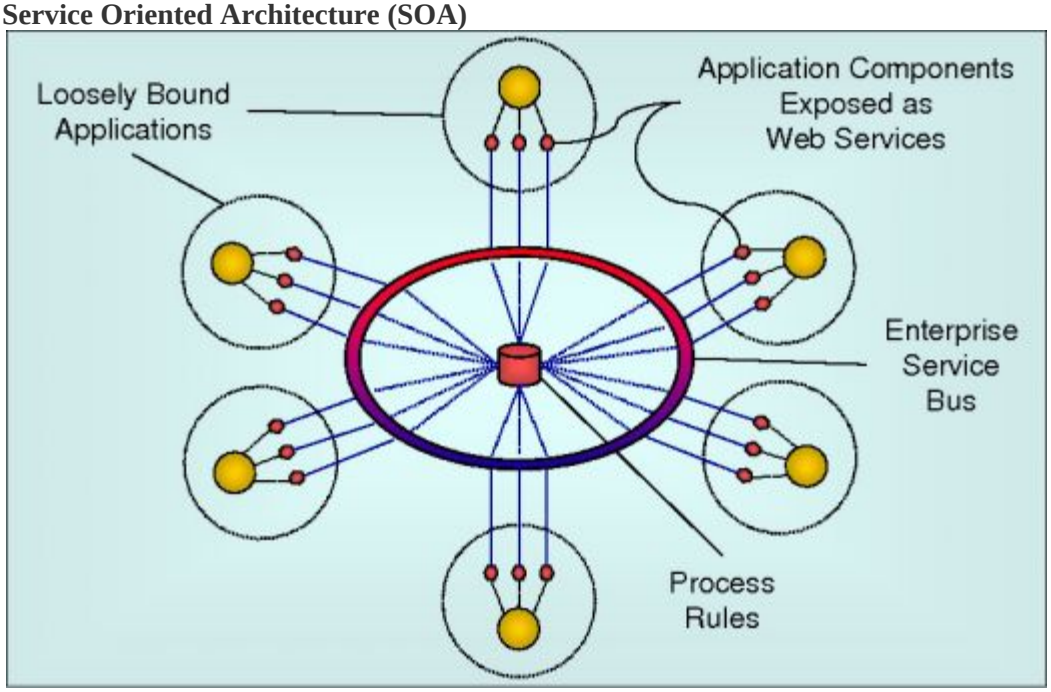


Fig. 2.4 Applications comprised of loosely bound Web Services

New developments that support Service Oriented Architecture

The breakthrough for SOA came with the acceptance of Web Services. Although CORBA, DCOM, and the like have been available for constructing SOAs for a number of years, the advent of Web Services was the first time that Microsoft and IBM could finally agree on a communication standard. They both wholeheartedly embrace Web Services.

After Web Services came Enterprise Service Bus (ESB) technology. Based on Web Services, and exhibiting all of the characteristics of the Messaging and BPM solutions previously supplied by the integration vendors, ESB has become the accepted standard for the creation of an organization's Service Oriented Architecture.

Without exception all of the integration vendors now provide an SOA architecture built on the concept of an Enterprise Service Bus (ESB).

Thanks to Web Services there's now a thriving integration industry that is creating more and improved services that can be used to construct bigger and better business solutions. There were initial concerns about

reliability and security but these have now been dealt with. There seems to be no stopping the universal acceptance of the Enterprise Service Bus and Service Oriented Architecture. When SOA implementation is guided by strategic business goals, chief benefits on an SOA are realized as follows: Alignment of IT with the business and maximal reuse of IT assets.

2.5 INTEGRATION WITHIN THE IMMIGRATION MINISTRY

This project considered the integration of services for the three departments within the ministry of immigration as described in Chapter 1.5 above. Each of these three ministerial departments was outlined below with their individual services enumerated, Ministry of Immigration and Registration of Persons (MIRP).

1. National Registration Bureau (NRB)

NRB enforces the Registration of Persons Act (Cap 107), Laws of Kenya, which provides for the compulsory registration and issuance of Identity Cards to all Kenyans who have attained the age of 18 years and above. The department has established six hundred registration centers countrywide and as it deems appropriate mounts Mobile registration units in areas without established offices.

The core functions and operations of the department are mandated through an Act of Parliament and they are outlined as:

- Identification and registration of all Kenyan citizens who have attained the age of eighteen (18) years and above.
- Production and issuance of secure identification documents
- Management of a comprehensive database of all registered persons
- Detection and prevention of illegal registration.

2. Immigration Department

Immigration Department is a security arm of the Government as well as a service department, charged with the responsibility of controlling entry and exit of persons seeking to live temporarily or permanently in Kenya. In discharging its functions under the Medium Term Plan (MTP) and Vision 2030 framework of “security of all persons and property throughout the Republic”, the department contributes towards security, national development and poverty reduction.

The Department derives its mandate from the Kenya Constitution Chapter VI, the Kenya Citizenship Act Cap. 170, the Immigration Act Cap.172, and the Aliens Restriction Act Cap.173 Laws of Kenya. The department is also guided by the Visa Regulations and international conventions e.g. Geneva Convention that Kenya is a signatory to. As stated in the mandates, the core functions of Immigration Department include:

- Formulation of national migration policy, regular review of Immigration Laws and regulations and advice to the government on national migration issues.

- Control and regulation of entry and exit of all persons at the country's airports, seaports and land border posts and the declaration and removal of prohibited immigrants.
- The issuance of Kenya passports and other travel documents including United Nations Travel Document (UNTD) in conjunction with the United Nations High Commissioner for Refugees (UNHCR).
- The control and regulation of residency through issuance and renewal of entry/work permits and other passes as provided by the Immigration Act, issuance of entry visas as provided under the Kenya Visa Regulations, the granting of Kenya citizenship to qualified foreigners under the Citizenship Act and the Kenya Constitution and the registration of all non-citizens resident in Kenya under the Aliens Restriction Act and Orders.
- Provision of consular services to nationals and foreigners at the Kenya missions abroad and the offering of quasi-consular functions to commonwealth countries who are not represented in Kenya and have requested the Kenya government for the service.
- The enforcement of the Immigration Act, the Citizenship Act, the Aliens Restriction Act, the Visa Regulations and the investigation and prosecution of persons who contravene these Laws and Regulation

3. Civil Registration Department

The Department of Civil Registration is the Government Agency charged with the responsibility of implementing the compulsory registration of all births and deaths occurring in Kenya irrespective of nationality. It also provides for the optional registration of the births and deaths of Kenya citizens occurring outside the country.

The functions and operations of the department are mandated through an Act of Parliament, the Births and Deaths Registration Act (Cap 149), Laws of Kenya and the Presidential Circular No 1 of 2008 on the organization of Government. As outlined in the mandates the core functions of Civil Registration department are:

- Registration of births and deaths
- Preservation, security and custody of births and deaths records.
- Issuance of births and deaths certificates
- Processing of vital statistics - both natality (birth statistics) and mortality (death statistics)
- Re-registration upon legitimating and recognition

A case for SOA in the Ministry

This study considered the use of the following applications by the three departments of the Ministry of Immigration to deliver different services to its citizens:

1) **Oracle e-Business suite** for citizen information management at the National Registration Bureau , call this *application A*

2) **A Java Application** that checks for fraud at the department of immigration - call this *application B*

3) **A Visual C#.NET application** for processing of vital statistics (both natality and mortality), preservation, security and custody of births and deaths records - call it *app C*

Suppose further that, application A has an Oracle database backend system, application B also runs on a separate Oracle database and application C runs on FLAT file based back end database system.

*The study further considered the process of a citizen online application for a passport at the immigration department. S*uppose the request application is an **APACHE /PHP application** executing on a separate hardware server. Call the *application D.*

To process this request:

1. **D** needs to query ***A*** for some minimal information regarding the citizen. It could be date of birth. **A** has to somehow communicate the results back to **D**. If all is well, go to step 2.

2. **D** needs to check that the request information supplied is free of fraud but unfortunately, it has to rely on the external application which is **B** in this case. Therefore **D** issues the fraud check request to app **B** with parameters such as "**Date of Birth".** B processes the request and returns the results back to **D**. In case of authenticity, D Proceeds to step3, processes the request for the citizen - we need to rely on **application** C**.** It sends the request to C**.**

3. **C**, processes the request by counter checking the citizen's vital statistics and ensures authenticity of the records and returns the result to the request system **D**.

Meanwhile, the citizen who issued the request via the request systems' web interface is waiting for the results. Having gotten the final response from **app C**, the request system **D** finally replies to the citizen via the web interface, perhaps saying "**Your request has been successfully processed.**

From this narrative:

1) To fulfill a single request process **different stepwise *sub processes*** have to be followed. Each sub-process is **fulfilled by a different component/application**.
2) The applications that fulfill these intermediate processes have to communicate when issuing requests and when conveying back the results of their computations on those requests.
3) Unfortunately, all the applications A, B, C and D are implemented on **different technologies, different. Naturally then, they can't communicate**.

The question that arises from the above narrative then becomes:

1) How then can these four processes A, B, C and D collaborate to fulfill a single request?

Solution A: Current situation

The traditional and most naïve way (which is also inefficient, error prone, unreliable and sometimes insecure) is:

Have a login staff access the new request from the system D, print out the request and have someone else key in the request into the citizen information system A and verify the details. Next, have the printed request manually checked for fraud by manually querying system B. If all is OK, the request is manually handed to the Civil Registration department to feed the request into the app A for verification processing. Once this is done, the final request voucher is given back to the login staff who updates the system and the results are made available to the citizen (requester).

Solution B: **Service Oriented *Architecture* (SOA).**

Have **a common unified communication framework/architecture** that makes it possible for different applications such as A, B, C and D above **to automatically communicate**, **share** (a minimal subset of)information in **a real time manner** to **fulfill a multistep, multi process regardless of differences in the hardware devices, operating systems, programming language models, communication protocols, data and representation formats**. In this case, all the processes are regarded as **services.** Services in this case are:

i) **Passport application processing service** (which is the end goal/primary service in relation to the citizen request for passport application) - the service facing the citizen or facing the end consumer. The service is fulfilled by application D.
ii) **Citizen Info service :** fulfilled by the citizen information management system A
iii) **Fraud Check service:** Fulfilled by the fraud management system B.
iv) **Verification service C** for verifying and the citizen's information processing

2.6 SERVICE ORIENTED ARCHITECTURE (SOA)

2.6.1 Formal Definition

Service Oriented Architecture (SOA) as a logical way of designing a software system to provide services to either end-user applications or other services distributed in a network through published and discoverable interfaces. (Papazolughu and Heuvel, 2008).

2.6.1.1 General Definition

Service orientation is generally defined as a **means for integrating across diverse systems**, African Journal of Business Management Vol. 4 2010 Mohammad Rehanand GoknurArzuAkyuz. Each IT resource whether an application, system or trading partner can be accessed as a **service.** These capabilities are available through **interfaces**; complexity arises when service providers differ in their operating system or communication protocols, resulting in inoperability. Service orientation uses standard protocols and conventional interfaces-usually Web services-to facilitate access to business logic and information among diverse services. Specifically, SOA allows the underlying service capabilities and interfaces to be composed into processes. **Each process is itself a service**, one that now offers up a new, aggregated capability. Because each new process is exposed through a standardized interface, the underlying implementation of the individual service providers is free to change without impacting how the service is consumed.

2.6.1.2 Definitions of SOA from different perspectives

i) **Business Analyst View Point:** A *set of services* that a business wants to expose to their customers and partners, or other portions of the organization.

ii) **IT Solutions Architect view point**. A set of architectural patterns such as *enterprise service bus*, *service composition*, and *service registry*, promoting principles such as *modularity*, *layering*, and *loose coupling* to achieve design goals such as separation of concerns, reuse, and flexibility.

iii) **Developers view point:** A *programming and deployment model* realized by standards, tools and technologies such as Web services and Service Component Architecture (SCA).

Note:

It is important to take a critical note from the start that SOA is not a product that can be purchased from a store. It is a way of structuring and integrating services and this can span over a period of time.

2.6.1.3 The Choice of Service Oriented Architecture

Complex and distributed IT resources are a concern for governments and businesses. Too frequently, the existing IT portfolio does not adequately meet specific business needs, is costly to manage and maintain, and is inflexible in the face of business growth and change. The solution, however, is not to rip and replace systems or applications, nor to completely renovate them, but rather to find a way to leverage existing IT investments so that overall organizational goals are effectively supported. Service orientation helps to accomplish these goals by making systems more responsive to business needs, simpler to develop, and easier to maintain and manage.

Implementing a solution architecture based upon service orientation helps any organizational set up plan ahead for change, rather than responding reactively to needs as they arise.

SOA provides the following benefits, (Erl, T., 2005)

- ***Reuse:*** **-** The ability to create services that are reusable in multiple applications.
- ***Efficiency:*** **-** The ability to quickly and easily create new services and new applications using combination of new and old services along with the ability to focus on the data to be shared rather than the implementation underneath.
- ***Loose technology coupling:*** **-** The ability to model services independently of their execution environment and create messages that can be sent any service.
- ***Division of responsibility:*** **-** The ability to more easily allow business people to concentrate on business issues, technical people to concentrate on technology issues and for groups to collaborate using the service contract.

2.6.2 Service Oriented Architecture (SOA) - Concepts

2.6.2.1 SOA Conceptual Model

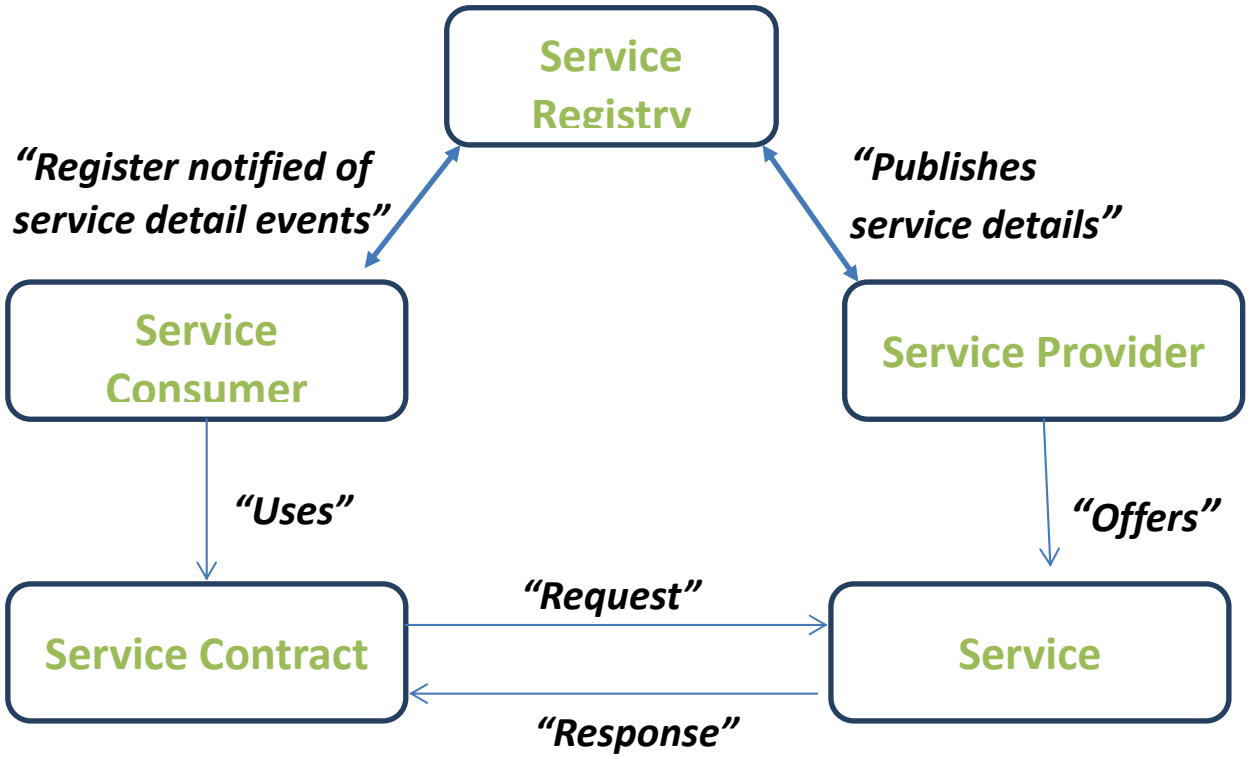


Fig. 2.5 SOA Conceptual Model

2.6.3 SOA Entities

(Michael, R., et al, 2008) defines various entities that go into a successful SOA platform. The "find, bind, and execute" paradigm allows the consumer of a service to ask a third-party registry for the service that matches its criteria. If the registry has such a service, it gives the consumer a contract and an endpoint address for the service. SOA consists of the following six entities configured together to support the find, bind, and execute paradigm.

2.6.3.1 Service Consumer

The service consumer is an application, service, or some other type of software module that requires a service. It is the entity that initiates the locating of the service in the registry, binding to the service over a transport, and executing the service function. The service consumer executes the service by sending it a request formatted according to the contract.

2.6.3.2 Service Provider

The service provider is the service, the network-addressable entity that accepts and executes requests from consumers. It can be a mainframe system, a component or some other type of software system that executes the service request. The service provider publishes its contract in the registry for access by service consumers.

2.6.3.3 Service Registry

A service registry is a network-based directory that contains available services. It is an entity that accepts and stores contracts from service providers and provides those contracts to interested service consumers.

2.6.3.4 Service Contract

A contract is a specification of the way a consumer of a service will interact with the provider of the service. It specifies the format of the request and response from the service. A service contract may require a set of pre-conditions and post-conditions. The preconditions and post conditions specify the state that the service must be into execute a particular function. The contract may also specify Quality of Service (QoS) levels. QoS levels are specifications for the non-functional aspects of the service.

For instance, a quality of service attribute is the amount of time it takes to execute a service method.

2.6.3.5 Service Proxy

The proxy design pattern (Gamma et al. 2002) states that the proxy is simply a local reference to a remote object. The service provider supplies a service proxy to the service consumer. The service consumer executes the request by calling an API function on the proxy. The service proxy, shown in *Figure 1*, finds a contract and a reference to the service provider in the registry. It then formats the request message and executes the request on behalf of the consumer. The service proxy is a convenience entity for the service consumer because it enhances performance by caching remote references and data. When a proxy caches a remote reference, subsequent service calls will not require additional registry calls. By storing service contracts locally, the consumer reduces the number of network hops required to execute the service.

For service methods that do not require service data, the entire method can be implemented locally in the proxy. If a method requires some small amount of service data, the proxy could download the small amount of data once and use it for subsequent method calls. The fact that the method is executed in the proxy rather than being sent to the service for execution is transparent to the service consumer.

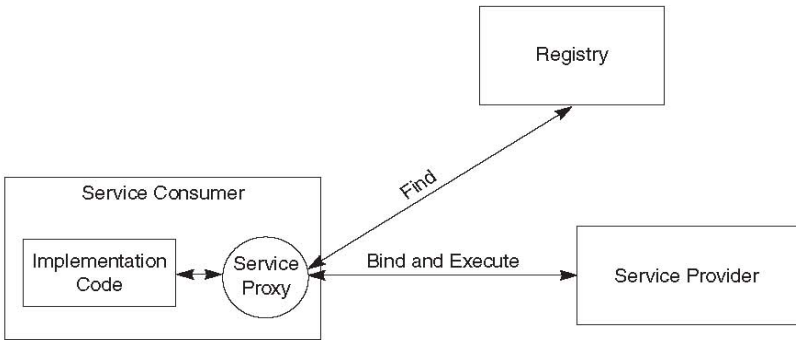


Fig. 2.6 A service Proxy

2.6.3.6 Service Lease

The service lease, which the registry grants the service consumer, specifies the amount of time the contract is valid: only from the time the consumer requests it from the registry to the time specified by the lease (Sun Microsystems, *Jini Technology Core Specification,* 2001). When the lease runs out, the consumer must request a new lease from the registry.

The lease is necessary for services that need to maintain state information about the binding between the consumer and provider. The lease defines the time for which the state may be maintained. It also further reduces the coupling between the service consumer and the service provider, by limiting the amount of time consumers and providers may be bound. Without the notion of a lease, a consumer could bind to a service

forever and never rebind to its contract again. This would have the effect of a much tighter coupling between the service consumer and the service provider.

With a service lease, if a producer needs to somehow change its implementation, it may do so when the leases held by the services consumers expire. The implementation can change without affecting the execution of the service consumers, because those consumers must request a new contract and lease. When the new contract and lease are obtained, they are not guaranteed to be identical to the previous ones. They might have changed, and it is the service consumer's responsibility to understand and handle this change.

2.6.4 SOA CHARACTERISTICS

Each system's software architecture reflects the different principles and set of tradeoffs used by the designers. Service-oriented architecture as a software architecture displays various characteristics (Bieber and Carpenter 2001, Stevens, 2002):

2.6.4.1 Discoverable and Dynamically Bound

SOA supports the concept of service discovery. A service consumer that needs a service discovers what service to use based on a set of criteria at runtime. The service consumer asks a registry for a service that fulfils its need.

A classic example best defines dynamic binding and discover; a banking application (consumer) asks a registry for all services that perform credit-card validation. The registry returns all entries that support this. The entries also contain information about the service, including transaction fees. The consumer selects the service (provider) from the list based on the lowest transaction fee.

Using a pointer from the registry entry, the consumer then binds to the provider of the credit card service. The description of the service consists of all the arguments necessary to execute the service. The consumer formats a request message with the data, based on the description provided by the directory pointer.

The consumer then binds the message to a transport type that the service expects and sends the service the request message over the transport. The service provider executes the credit-card validation and returns a message, whose format is also specified by the service description. The only dependency between producer and consumer is the contract, which the third-party registry provides. The dependency is a runtime dependency and not a compile-time dependency. All the information the consumer needs about the service is obtained and used at runtime.

This example shows how consumers execute services dynamically. Clients do not need any compile-time information about the service. The service interfaces are discovered dynamically, and messages are constructed dynamically. The removal of compile-time dependencies improves maintainability, because consumers do not need a new interface binding every time the interface changes.

2.6.4.2 Self-Contained and Modular

One of the most important aspects of SOA is the concept of modularity. A service supports a set of interfaces. These interfaces should be cohesive, meaning that they should all relate to each other in the context of a module. The principles of modularity should be adhered to in designing the services that support

an application so that services can easily be aggregated into an application with a few well known dependencies. Since this is such an important concept when creating services, I will explain some of the principles of modularity and, in particular, how they apply to the creation of services.

2.6.4.3 Modular Decomposability

The *modular decomposability* of a service refers to the breaking of an application into many smaller modules. Each module is responsible for a single, distinct function within an application. This is sometimes referred to as "top-down design," in which the bigger problems are iteratively decomposed into smaller problems. For instance, a banking application is broken down into a savings account service, checking account service, and customer service. The main goal of decomposability is reusability.

2.6.4.4 Modular Understandability

The *modular understandability* of a service is the ability of a person to understand the function of the service without having any knowledge of other services. For instance, if a banking application implements a checking account service that does not implement a deposit function but instead relies on the client to use a separate deposit service; this would detract from the service's modular understandability. The modular understandability of a service can also be limited if the service supports more than one distinct business concept.

2.6.4.5 Modular Continuity

The impact of a change in one service requiring a change in other services or in the consumers of the service. An interface that does not sufficiently hide the implementation details of the service creates a domino effect when changes are needed. It will require changes to other services and applications that use the service when the internal implementation of the service changes. Every service must hide information about its internal deign. A service that exposes this information will limit its modular continuity, because an internal design decision is exposed through the interface.

2.6.4.6 Modular Protection

The *modular protection* of a service is sufficient if an abnormal condition in the service does not cascade to other services or consumers. For instance, if an error in the checking account service causes invalid data to be stored on a database, this could impact the operation of other services using the same tables for their data. Faults in the operation of a service must not impact the operation of a client or other service or the state of their internal data or otherwise break the contract with service consumers. Therefore, ensure that faults do not cascade from the service to other services or consumers.

2.6.4.7 Direct Mapping

A service should map to a distinct problem domain function. During the process of understanding the problem domain and creating a solution, the designer should create boundaries around service interfaces that map to a distinct area of the problem domain. This is important so that the designer creates a self-contained and independent module. For instance, interfaces that deposit, withdraw, and transfer from a checking

account should map to the checking account service.

2.6.4.8 Loose Coupling

Service-oriented architecture like every software architecture, strives to achieve loose coupling between modules between service consumers and service providers and the idea of a few well-known dependencies between consumers and providers.

A system's degree of coupling directly affects its modifiability. If the consumer of the service does not need detailed knowledge of the service before invoking it, the consumer and provider are more loosely coupled.

SOA accomplishes loose coupling through the use of contracts and bindings. A consumer asks a third-party registry for information about the type of service it wishes to use. The registry returns all the services it has available that match the consumer's criteria.

2.6.5 SOA Entry Points

The five entry points defined by IBM - based upon real customer experiences – helps any business to benefit by implementing predefined SOA solutions. These entry points are driven by both:

- Business needs *(people, process, and information entry points)* and,
- IT needs *(connectivity and reuse entry points).*

Here are the descriptions of the five entry points:

People: Focuses on the user experience to help generate innovation and greater collaboration, which enables consistent human and process interaction, thus improving business productivity. Using SOA you can, for example, create service-based port to increase this collaboration.

Process: Helps companies know what is happening in their business, allowing them to improve existing business models. SOA transforms business processes into reusable and flexible services, allowing for the improvement and the optimization of these new processes.

Information: Using this entry point to SOA leverages information in the organization in a consistent and visible way. By providing this consistent and trusted information throughout all areas of the business, all areas of your company are empowered to innovate, thus allowing for a more effective competition. SOA gives a better control over information, and by aligning information with business processes, new relationships interesting relationships are discovered.

Connectivity: Connectivity effectively connects infrastructure, integrating all of the people, processes, and information in the organization. By having flexible SOA connections between services, and throughout the entire environment, an existing business process can be delivered without much effort through a different business channel. Connection to external partners outside the firewall is achieved in a secure way.

Reuse: Reusing services with SOA allows for tapping into the services that already exist in the organization. By building from existing resources, business processes are streamlined to ensure consistency and cut

development time. Duplication of functionality in the services can be drastically reduced and get to take advantage of using the proven core applications the people in the company are familiar with.

2.6.6 SOA Deliverables

At the end of thevproject, SOA ensures that its implementation setting (Roch, E. 2006):

- Implements a secure, scalable, and flexible IT architecture that best positions it to take advantage of new technologies and accommodate future growth within the ministry of state immigration and registration of persons.
- Migrates legacy data within the ministry from retiring applications into modernized ones, including customized applications.
- Profiles legacy data and develop data quality plans to cleanse and standardize data as it is moved into new applications
- Converts data warehouse feeds from legacy applications to new applications.
- Assists in the operations and maintenance of parallel environments as modernizations are tested through to deployment.
- Protects and maintains data security and fire walled environments for sensitive deployments.

2.6.7 SOA IMPLEMENTATION TECHNOLOGY

2.6.7.1 Service Orientation via Web - services

For the sake of distinguishing a web service from a web application, which is a commonly asked question, a web service is a non-browser, non-operating system independent message based design implemented via technologies such as WSDL, UDDI, XML and SOAP (Rouse, 2007).

A web application on the other hand is a simple (GUI) application designed to run in a specific browser on a specific operating system (Rouse, 2011). Therefore by using web services, different applications such as the departmental applications in the illustration above can communicate freely without regard to the underlying technology. In this case all information is exchanged using a given web service standard. The standard could be one of the ones mentioned herein.

In the SOA architecture, entities (applications) that **initiate requests** and those that **service the requests**, the former apps could be regarded as **"client applications**" and the latter apps could be regarding as being relatively "**server applications".** Therefore implementing SOA using any of the mentioned standards, any pair of the applications is relatively **client and server.**

2.6.7.2 WEB SERVICES TECHNOLOGIES

2.6.7.2.1 Universal Description, Discovery and Integration (UDDI)

Universal Description, Discovery and Integration (UDDI) is a web based distributed directory like traditional phone book's yellow and white pages that enables businesses to list themselves on the Internet and discover

each other. It defines a registry service - a Web service that manages information about service providers, service implementations, and service metadata - for Web services and for other electronic and non-electronic services.

The service providers can use UDDI to advertise the services they offer while service consumers can use UDDI to discover services.

2.6.7.2.2 Web Services Description Language (WSDL)

The WSDL refers to Web Services Description Language, is an XML based protocol used for sending and receiving the information through decentralized and distributed environments. WSDL is an integral part of UDDI that was developed jointly by Microsoft and IBM.

It defines what services are available in its Web service and also defines the methods, parameter names, parameter data types, and return data types for the Web service. The WSDL document is quite reliable and applications that use web services accept it

2.6.7.2.3 SOAP (Simple Object Access Protocol)

SOAP is an XML-based messaging protocol that defines a set of rules for structuring messages that can be used for simple one-way messaging but is particularly useful for performing RPC-style (Remote Procedure Call) request-response dialogues (Webopedia). It is not tied to any particular transport protocol though HTTP is popular. Nor is it tied to any particular operating system or programming language so theoretically the clients and servers in these dialogues can be running on any platform and written in any language as long as they can formulate and understand SOAP messages. As such it is an important building block for developing distributed applications that exploit functionality published as services over an intranet or the internet.

2.6.7.2.4 Reasons for SOAP

SOAP (Simple Object Access Protocol) is given an upper hand because of its agility and simplicity, as the most popular technology for concretizing web services and consequently the most preferred method for implementing service oriented architecture, (Sommers, F., 2003). The standard **uses HTTP as the transport mechanism at the application layer and XML as the data format.**

Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.

A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

- Using SOAP over HTTP allows for easier communication through proxies and firewalls than previous remote execution technology.
- SOAP is versatile enough to allow for the use of different transport protocols. The standard stacks use HTTP as a transport protocol, but other protocols are also usable (e.g. SMTP, RSS).
- SOAP is platform independent.
- SOAP is language independent, simple and extensible.

2.6.7.2.5 SOAP with Attachments API for Java

The SOAP with Attachments API for Java (SAAJ) provides a standard way to send XML documents over the Internet from the Java platform.
SAAJ enables developers to produce and consume messages conforming to the SOAP 1.1 specification and SOAP with Attachments note.
SOAP messaging applications can also be used directly instead of using JAX-RPC or JAX-WS.

2.6.7.2.6 A summary of web services technologies - Java SOAP, XML/RPC

The Java API for XML Web Services (JAX-WS): It is a Java programming language API for creating web services. It is part of the Java EE platform from Sun Microsystems. Like the other Java EE APIs, JAX-WS uses annotations, introduced in Java SE 5, to simplify the development and deployment of web service clients and endpoints. It is part of the Java Web Services Development Pack.

2.6.8 SOA IMPLEMENTATION APPROACHES

Three main approaches to consider for the implementation of Service Oriented as described by Aranjani, A., 2004, are outlined below:

- **Top Down approach**

This approach takes a broader, more enterprise perspective and more strategic point of view. It is concerned with the overall set of enterprise requirements now and overtime.

- **Bottom Up approach**

This approach starts from a perspective of existing systems, technology and common services. It is concerned typically with specific projects the immediate requirements.

- **Middle-out approach**

This approach takes the best of both of the above approaches. It is concerned with producing both high end business and information architecture and design artifacts and working deployed services.

2.7 SOA METHODOLOGIES

2.7.1 Thomas Erl's Mainstream Service Oriented Architecture Methodology (MSOAM)

MSOAM is focused on both the definition and delivery of collections of services called "service inventories" and the definition and delivery of individual services as they progress through the following individual lifecycle stages:

• Business Modeling/Define Enterprise Business Models
• Service-Oriented Analysis/Inventory Analysis
• Service Contract Design (Service-Oriented Design)
• Service Logic Design (Service-Oriented Design)
• Service Development
• Service Testing
• Service Deployment
• Service Governance

MSOAM is a practical approach to SOA, but it expects that business modeling tasks be completed prior to proceeding with service-specific phases. Therefore, it provides no real guidance in this area. This gap can be filled by RUP Business Modeling.

There are two entry points for the marriage of RUP and MSOAM, each of which represents a different level of business modeling:

- Step 1: preparatory and on-going business modeling for the Define Enterprise Business Models stage
- Step 2: Iterative and system model-centric business modeling for the Service-Oriented Analysis stage

2.7.1.1 Defining Analysis Scope/RUP Business Modeling

MSOAM requires that this step be completed so that the scope of the analysis moving forward is well defined. However, as a result of the work performed by the previously explained preparatory RUP Business Modeling steps, the analysis scope will already have been determined.

2.7.1.2 Identify Affected Systems

This traditional Service-Oriented Analysis step requires that before proceeding to the service modeling stage where service candidates are defined, architects take a good look at the systems that relate to the analysis scope. Any constraints or features they provide that may tie into the modeling and design of services need to be documented as input for the service modeling sub-process.

2.7.1.3 Define System Model

As a result of combining RUP with MSOAM, this new step is inserted here in order to make a transition from a business process to an "automation process".

In the preceding RUP Business Modeling phases the focus was on the business use-cases and related business elements in relatively abstract terms. Now, the focus is on how the business process needs to be automated. This is documented as part of a system model specification.

2.7.2 Michael Rosen Et Al SOA Implementation Methodology

This is a high-level methodology for implementing enterprise SOA solutions outlined and as presented by Michael Rosen *et al* in the book; ***Applied SOA, Service Oriented Architecture and Design Strategies***. This methodology consists of the following major activities:

SOA reference architecture – This step define the important aspects of the SOA reference architecture, in particular what a service is, the types of services and their relationships, design and implementation concepts and processes, and relationships too their architectures and communications.

Business architecture definition - The first step is to define the enterprise business architecture. This influences the processes, services, information, and enterprise solutions that will be built.

Service identification - Define a set of services within the enterprise context that supports the business architecture. The overall set of services makes up the service inventory.

Semantic information model definition - During this step, an enterprise information model that defines the shared semantics of processes and services is created. This activity is often done in parallel with service identification.

Service specification - At this step, service contracts that can be used at design time for the selection of appropriate services in solutions are created. The service specification includes the service interface as well as other usage and dependency information.

Service realization - Design and implement services.

Implementation of service-oriented solutions - At this step**,** an enterprise solution from services is built by first creating a high-level business architecture and service inventory. The first set of services to support specific business goals is then implemented.

2.7.3 IBM's Service Oriented Modeling Architecture - (SOMA)

SOMA is a software development life-cycle method invented and initially developed in IBM for designing and building SOA-based solutions. This method defines key techniques and provides prescriptive tasks and detailed normative guidance following the prescriptive steps:

- **Business modeling and transformation**

In this phase, the business is modeled, simulated, and optimized, and a focal area for transformation that will drive a series of subsequent projects is identified.

- **Identification phase**

The identification phase pertains to the identification of the three fundamental constructs of SOA: services, components, and flows. This is achieved by a set of complementary service identification techniques.

- **Specification Phase**

High-level design as well as significant parts of the detailed design of service components is completed in this phase in order to design SOA.

- **Realization phase**

Realization decisions are validated during this phase by performing technical feasibility exploration that seeks to exercise the architectural decisions. A key task in realization is to detail and instantiate the SOA reference architecture.

- **Implementation and deployment, monitoring, and management phases**

In the implementation phase, the service, functional, and technical components that realize services, components, and flows are constructed, generated and assembled.

2.7.4 An Evaluation of SOA Methodologies

The table below *(Table 2.1)* indicates an evaluation of some of the most influential SOA methodologies. Despite their differences though, all of these approaches share a service oriented mentality, with the purpose of lessening the issues of clients and companies, students and teachers, citizens and government employees alike.

SOA METHODOLOGY	IBM's SOMA Evaluation	Erl's MSOAM Evaluation	Michael Rosen *et al* Evaluation
Delivery strategy	Employs the meet-in-the middle strategy.	Top - Down approach	Bottom - up approach
Business/Process Agility	High	Medium	Medium
Industrial application	Extensive industrial application.	Not yet	Case studies available.
Proprietary	Available. Documentation is available openly for interested parties.	Not openly available. Detailed specifications are not openly available. It is difficult to fully analyze.	No proprietary issues. Full documentation openly available and accessible.
Tools support/ Manageability of complexity	Techniques such as GSM, SLT determine the granularity of service.	MSOAM does not have tools available for use in coherence with the methodology.	Enterprise IT decomposition.
Adoption of existing processes/techniques/ Notation	Partially on RUP (Rational Unified Process) and Business Processing Modeling.	No technique is defined for the architectural design.	Unified Modeling Language (UML)
Supported roles	Development is declarative and business process oriented through service composition	Development is programmatic and component.	Development is declarative and business process oriented through service composition

Table 2.1: An evaluation of current SOA Methodologies.

Analysis and design techniques for service-oriented development and integration is outlined by the IBM's Service Oriented Modeling Architecture (SOMA) methodology. It is a full-blown modeling methodology that refers to the more general domain of service modeling necessary to design and create SOA.

2.8 SOA CASE STUDIES

Mapping of SOA oriented components are discussed through two case studies; Indian and Tunisian context to appreciate the architecture. .

2.8.1 Case Study 1: Understanding SOA Perspective of e-Governance in Indian Context

Service Oriented Architecture (SOA) is a contemporary phenomenon which is targeted for efficient and inclusive business automation. SOA principles and models are being used for building good e-government systems. Good results of e-government systems notwithstanding, such projects are however not free from challenges in many countries. E-governance projects have fallen short of citizen expectations in developing countries (Mehdi, 2005). In Indian context, this issue is relevant as it poses a major challenge for e-

governance policy makers to successfully incorporate citizen participation, especially with development perspectives and sustain this participation during scale up.

E-governance projects in India need to follow SOA principles in order to make them successful in terms of sustainability, providing appropriate services to citizens. Essentially, SOA principles provide such ambience to extend desired support to e-governance initiatives in India.
A scenario is built through SOA architecture to showcase the possible effect of SOA principles in order to appreciate citizen centric services taking scale-up issues into consideration. This framework is discussed to reflect the underpinnings of orchestration of services on demand and service provisioning through e-governance initiatives for effective implementation SOA principles.

2.8.1.1 Indian E-Governance Systems: Development Perspectives

Interoperability has been a critical evaluation criterion for enabling interstate transactions, managing information flow seamlessly and overseeing the backend process for effective delivery of citizen services.

In India, e-governance system is still evolving and is not free from challenges as experienced in global terms. There are however many successful ICT initiatives in India oriented towards rural development with a focus to address some specific issues of rural citizens, thus forming “islands”. The aim is to provide a portfolio of services to the citizens integrated with e-governance backbone to install a good e-governance system without getting affected during scale up phase. Service orientation is an essential component for Indian e-governance systems since 'desired services' need to be provided to the citizens. This is possible through a SOA based model which would enable service orientation through citizen demands. Aggregated service demanded are the inputs for the 'service provisioning agencies' in the national network engaged for establishing the orchestrated link to manage the 'service brokering' facility and supply the services. This provides a scope for the citizens to receive the desired service through SOA based service model.

2.8.1.2 SOA Based E-Governance Strategy

SOA principles draw strength from the benefits of well practiced architectures in software engineering discipline like client-server, distributed (including component object (COM)/distributed component object (DCOM) and Object-Oriented) architecture. (Erl T., 2008).
SOA builds on the strengths of 'application architecture' and 'enterprise architecture' and therefore, has potential to manage e-governance projects. SOA is expected to provide 'universal service identifier' in the system so that desired service can be identified 'on demand' with least transaction time, transaction cost and independent of spatial constraints.

It is seen that various service components of SOA can contribute to the Indian e-governance system in order to provided desired services. The components are 'citizen demand on services', 'service on demand aggregation', 'service-on-supply aggregation', 'service orchestrators' and 'service providers'. A seamless

integration of all the services and service provisioning components need to collaborate effectively to focus on citizen centric services. In other words, there are concurrent attempts to provision citizen centric services taken by central and state authorities. Of late, central administration has deployed mission mode projects with states collaborating as part of NeGP (Chandrashekhar, 2006). Therefore, convergence of services is of prime importance so as to provide commercial approach to the services and establish sustainable and remunerative information service provisioning.

2.8.1.3 Conclusion

SOA architecture based treatment to the Indian e-governance therefore reveal that there is a need to carefully conceptualize and to incorporate all the characteristics of SOA in order to provide citizen centric services. It is far more important that countries like India need to carefully articulate services with active collaboration of the citizens in order to deliver good governance systems.

2.8.2 Case Study 2: A Service Oriented Product Line Architecture for E-Government in Tunisia

The successful establishment of an E-Government system is certainly the result of a good design of its software architecture. Architecture should offer reusability as an essential characteristic in the development of governmental applications. However, this reusability, although recommended by the standards, can be more profitable by the adoption of a systematic, large scale reuse approach.

The application of the SOPL (Service Oriented Product Line) approach promises improvements in productivity, time-to-market, quality, and cost. This is a layered approach for the architecture model for E-Government proposed a representative sample consisting of three layers of E-Government architectures was studied.

The Front-end services layer: The Front-end represents the user interface of the E-Government system. This layer represents a portal including all the governmental services. This portal constitutes a single access point via the Web to the services intended for the users of this system.

The Back-end services layer: This layer encapsulates various workflow applications responsible for the execution of the workflows materializing the different services offered by the organization.

The legacy systems layer: The legacy systems layer represents the various information systems already implanted within the governmental organizations connected with Intranet networks.

2.8.2.1 The SOPL approach

SOPL as a recently introduced approach based on the concepts of SOA which offers an answer to the problems of heterogeneity and interoperability of systems. Nevertheless, this architecture does not take into account the changes which can occur for the services. Moreover, it does not have the necessary mechanisms for the identification of the services in the suitable level of granularity. This has led the research community in the area of software reuse to opt for the combination of SOA approach with PLE (Product Line Engineering).

To introduce the SOPL approach, the focus was on the work of Medeiros & al [14] who presented the life cycle of a service line consisting of two distinct phases:

Phase1: The domain engineering: Represents the development for reuse by first, identifying components, services and composite services candidates for reuse. A study of a range of governmental services offered by the Tunisian Ministry of the interior and local development as the demand of National Identity Card (CIN), Passport and Bulletin n°3.

Phase2: The application engineering represents the development by reuse. This phase selects the components, services and composite services specific to a product. For the Tunisian case study, we have three products to derive namely the "CIN", the "Passport" and the "Bulletin n°3" workflow applications. For each one of these products, the three steps of the domain engineering phase of the SOPL life cycle were applied.

2.8.2.2 Conclusion

The point of interest in this work concerns the software architecture of an E-Government system presented by the proposed architectural model detailing its phases while being particularly interested in the back-end services layer. The option for the application of a systematic, large scale reuse approach for the production of these services enhanced reuse with different granularities, permit a better time to market specifically when faced to frequent changes in government laws, hence the need for new or adaptable e-government services (software applications). SOPL approach was adopted for its hybrid advantage of combining SOA architecture and Product Line Architecture. This choice is motivated by the need for reuse and interoperability additionally with other quality attributes for E-Government architecture such as usability, scalability, security, transparency, legality, symmetry and responsibility. We have applied the SOPL life cycle on a case study in the domain of E-Government, a domain characterized by business processes sharing similarities, in order to test its feasibility.

CHAPTER THREE

METHODOLOGY

3.1 INTRODUCTION

This chapter describes an in-depth step by step methodology that was employed for SOA implementation at Ministry of Immigration and Registrar of Persons (MIRP).

3.2 SERVICE ORIENTED MODELING ARCHITECTURE (SOMA)

SOMA consists of three major phases of identification, specification and realization of the three main elements of SOA, namely, services, components that realize those services and flows that can be used to compose services. It builds on current techniques in areas such as domain analysis, functional areas grouping, process modeling, component-based development, object-oriented analysis and design and use case modeling.

Benefits of SOMA over the other methodologies;

The following are the distinctive characteristics of SOMA that were essential for the implementation of SOA at the MIRP:

Delivery strategy: SOMA employs the meet-in-the middle strategy which finds a balance between incorporating service-oriented design principles into business analysis environments without having to wait before integrating Web services technologies into technical environments.

Lifecycle coverage: SOMA approach aims to support the full SOA lifecycle, including planning, analysis and design, construction, testing, deployment, and governance activities, while others limit their scope to a subset of these phases, such as analysis and design.

Granularity of services: SOMA introduces new techniques such as goal-service modeling, service model creation and a service litmus test to help determine the granularity of a service.

Availability: SOMA like a number of methodologies, proposed by industry players has its documentation available to the interested public, making it easier to fully analyze its capabilities and to make comparisons.

Business/Process agility: SOMA methodology suggests an agile approach to Service Oriented development in order to address risks and add flexibility to change to both the business and processes.

Adoption of existing processes/techniques/notation: A large number of SOA methodologies propose reusing proven existing processes. SOMA appreciates service-oriented development as an evolutionary rather than revolutionary step in software engineering.

Industrial application: It is important that a methodology be validated in proof-of-concept case studies to show that it has practical applicability and to refine it based on feedback from the case studies. SOMA is one of the existing SOA methodologies that has an extensive industrial application in. The expanse of these projects is wide and deep, ranging from diverse fields such as health care, telecommunications and financial services to public-sector implementation projects.

3.3 SOMA PHASES

It is important to note that SOA implementation as a process is iterative and incremental where high-level business architecture and service inventory are created first, and then followed by the implementation of the first set of services to support specific business requirements that will eventually realize the optimum business goals.

3.3.1 Business Modeling and Transformation

In this phase, the business is modeled, simulated, and optimized, and a focal area for transformation is identified that drives a series of subsequent projects using the set of phases shown in SOMA high level diagram below and discussed in the subsequent sections. Note that this phase is not strictly required but is highly recommended.

Service Oriented Modeling Architecture (SOMA) Phases

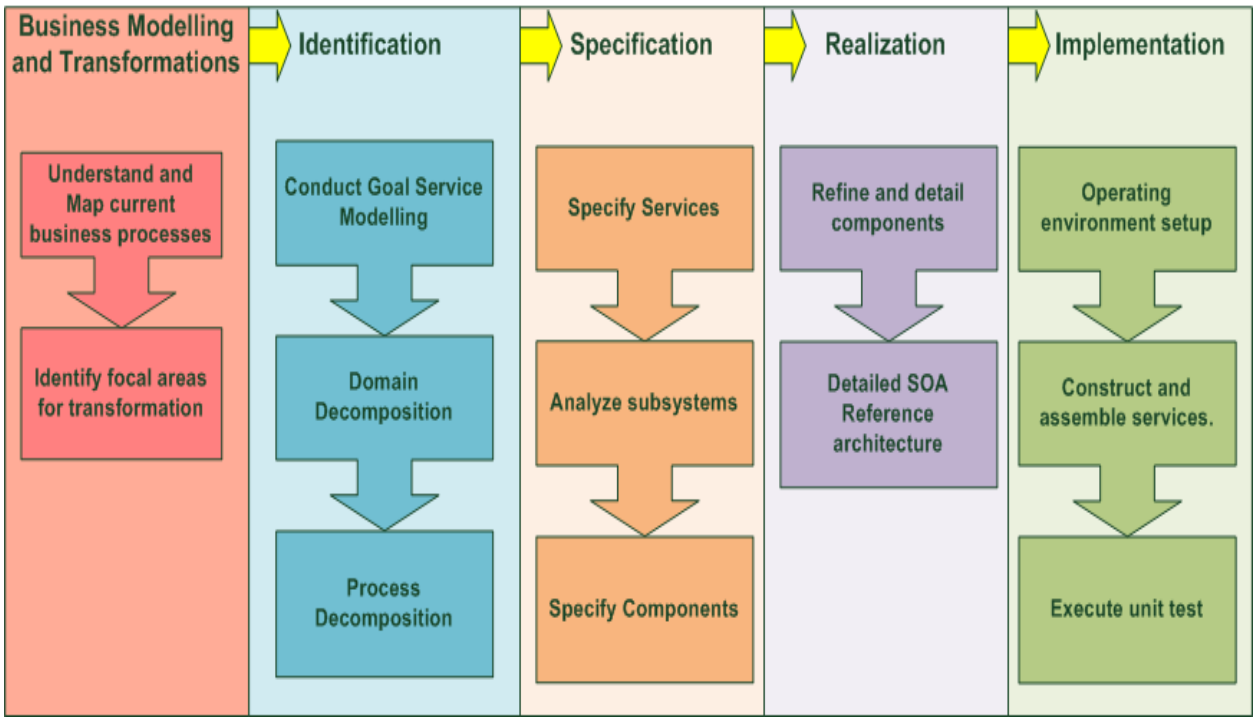


Fig. 3.1 High-level SOMA diagram

SOA solutions are hybrid in nature and typically include multiple solution types. This is because services identified and specifies during the early phases in SOMA can be realized in subsequent phases of SOMA by different scenarios, such as custom development, legacy integration and transformation, and package application integration.

3.3.2 Identification Phase

The identification phase pertains to the identification of the three fundamental constructs of SOA: services, components and flows. It is a best practice to utilize a set of complementary service identification techniques since relying on a single technique tends to create an incomplete set of services. This introduces information entropy early in the development life cycle, often to be remedied at greater cost later in the service life cycle as it entails greater efforts of service refactoring. Additionally, this often leads to the failure to identify service dependencies early on, which impacts release planning and, ultimately, project delivery.

3.3.3 Specification Phase

In the SOMA specification phase, the SOA is designed. High-level design as well as significant parts of the detailed design of service components is completed in this phase. During the specification phase, existing assets are leveraged and the services, flows and components from the identification phase are further elaborated in an iterative and incremental fashion to help reach the realization decision. The service model is also elaborated in terms of service dependencies, flows and composition, events, rules and policies, operations, messages, nonfunctional requirements, and state management decisions.

Two foundational and preparatory activities are performed during this phase:

a) Information models are elaborated and specified.

b) A fine-grained analysis and specification of existing assets.

This is a summary of the core activities during the specification phase;

i. After the business entities and their high-level structure and relationships (Entity Relationship Diagrams are relevant to the service scope) are identified during the identification phase, the conceptual data model is elaborated into a logical data model to populate the attributes to be implemented, which are defined in terms of their domains or logical data types.

ii. Design of service messages which include input, output, and error messages. In order to eliminate multiple data transformations in the service layer, a best practice is to use a common message model that defines the message flow in the services layer by selecting of a message format (e.g., Extensible Markup Language) define the set of types, elements, and attributes representing the business entities and their business attributes.

iii. An analysis of system interfaces and the input and output parameters of the existing systems is done in order to perform a fine-grained mapping of the service to a specific legacy application transaction. The mapping provides a potential set of realization decisions for the SOA.

iv. Identification of duplicate, unstructured, and unused code of a functionality to be exposed as a shared service across channels.

3.3.4 Service Specification Phase

Service specification is the core of the service modeling activity and focuses on elaborating the detailed design of the services.

3.3.5 Subsystem Analysis

Subsystems signify logical IT boundaries for business functionality. They are identified by functional decomposition of areas

3.3.6 Component Analysis

Patterns that help in structuring service components into a set of functional components are explored. Also considered are the technical components responsible for providing auxiliary support from both the technological and infrastructure perspective.

3.3.7 Realization Phase

Realization decisions are validated during this phase by performing technical feasibility exploration that seeks to exercise the architectural decisions and highest project risk factors through extensible prototypes designed and developed early on.

Corresponding pattern categories are selected to handle several problem domains e.g.

- **a)** Information service patterns for information realization,
- **b)** Enterprise Service Bus (ESB) patterns for integration scenarios,
- **c)** Rule patterns for rule realization.

3.3.8 Technical Feasibility Exploration And Realization Decisions

In technical feasibility exploration, risk factors and technical challenges focusing on nonfunctional requirements are planned for

A key task in realization is to detail and instantiate the SOA reference architecture. Technical feasibility exploration is conducted for each of the service interaction patterns chosen in the solution based on layers of the SOA reference architecture.

Technical feasibility exploration designates interaction patterns that are selected based on several factors, including the maturity of the organization adopting SOMA and its underlying technological infrastructure.

3.3.9 SOA reference architecture layers

SOA reference architecture provides a snapshot view of SOA. This view facilitates communication and provides a representation of progress and evolution of the SOA in one high-level diagram.

3.3.10 Implementation and Deployment, Monitoring, and Management Phases

In the implementation phase, the service, functional, and technical components that realize services, components, and flows are constructed, generated and assembled. Necessary wrappers or other mechanism by which an existing component can participate in realizing a service are also created.

Also to be performed in this phase are unit, integration, and system testing for services, components, and flows.

SOMA uses a variety of solution templates that include support for custom development, packaged application integration, legacy integration and transformation, ESB design and use, and composite business services.
Activities and tasks from these solution templates are extended for different solution types in the overall solution.

In the deployment, monitoring and management phase, the focus is on packaging, provisioning, executing user-acceptance testing and deployment of services.

This phase is also extended by activities and in the overall solution. SOMA provides support of monitoring and management of business processes and performance monitoring in the production environment.

CHAPTER FOUR

SOA ANALYSIS AND DESIGN AND IMPLEMENTATION

The SOA system was designed and implemented using the methodology steps described in chapter 3.3 above ***(IBM's Service Oriented Modeling and architecture (SOMA))***

4.1 BUSINESS MODELING AND TRANSFORMATIONS

Ministry of Immigration and Registration of Persons (MIRP) offers three major lines of services to the citizens: Issuance of passport, issuance of National Identification cards, and issuance of birth and death certificates. This is made possible through these three custodial enterprises respectively; Immigration department, National Registration Bureau and Civil Registration Department.

As part of the systems analysis carried out at the MIRP, I interviewed relevant officers in these key departments who confirmed that the Ministry of Immigration and Registration of Persons (MIRP) uses spreadsheets for managing their daily transactions.

References and Sources of Information

1) Senior ICT Officer, Immigration Department, 3rd floor Nyayo house. "Oral Interviews held **8th June, 2012**
2) Assistant Manager, Training and Research, Immigration Department, 8th floor, Nyayo house, "Oral Interviews held between 4**th and 11th June, 2012**
3) Senior Population Statistician, Civil Registration Department, Ground floor Hass Plaza, Upper Hill area Nairobi, "Oral Interviews held between **4th and 8th June, 2012.**
4) Senior Assistant Director - Field Services**,** National Registration Bureau, 8th floor NSSF building, Nairobi, "Oral Interviews held between **7th May and 18th June, 2012**

4.1.1 Business Flow Charts

The following charts indicate the processes that go into ensuring that each of the above services are delivered to the citizens.

Passport Application process

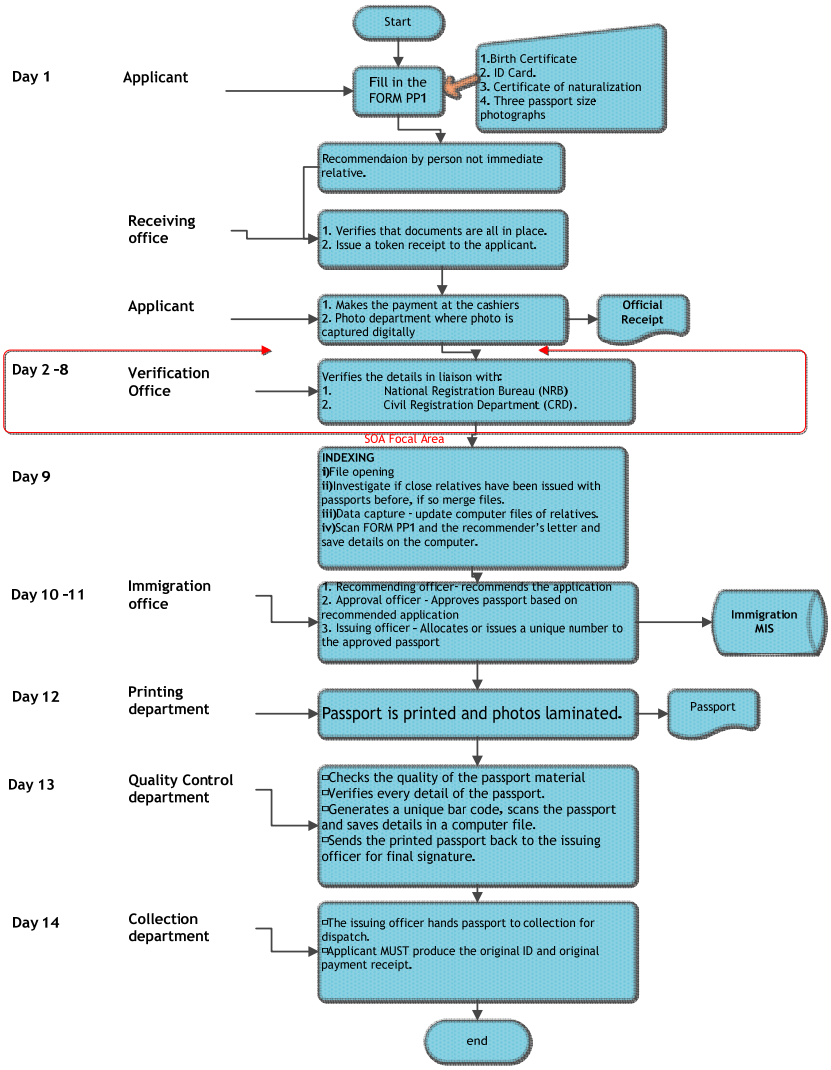
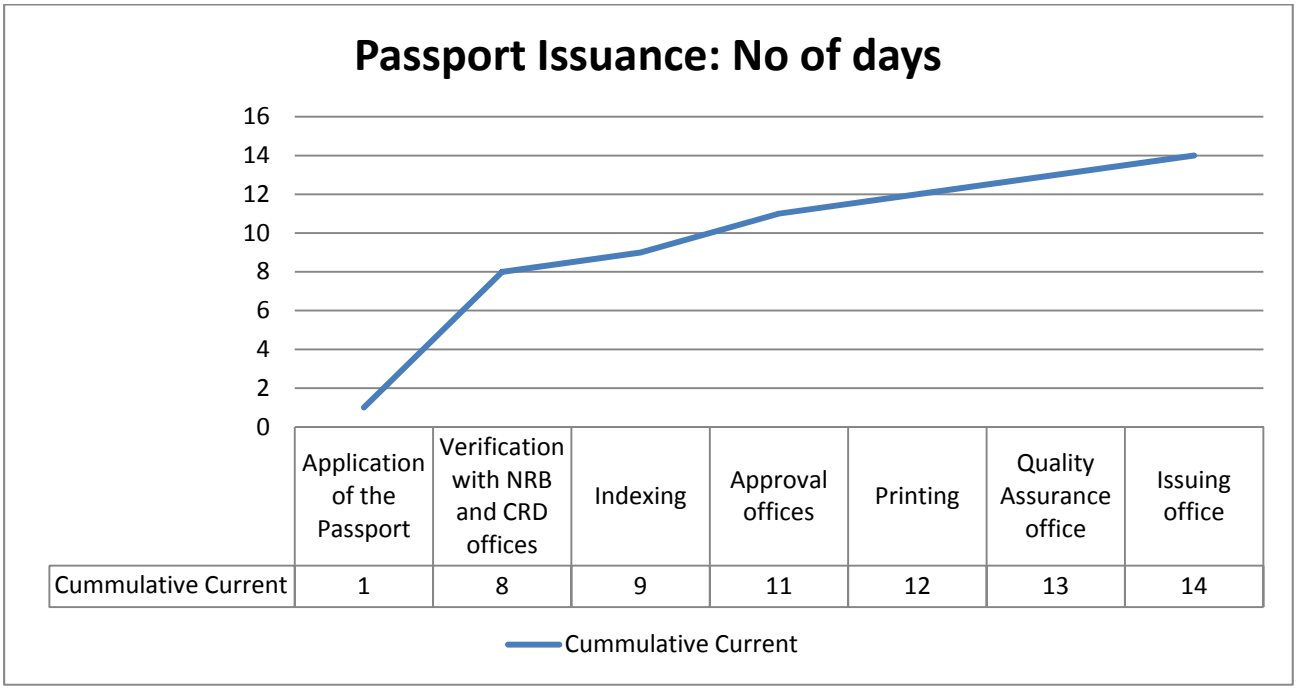


Chart 4.1 Passport application/Issuance process



Graph 4.1 Passport application/Issuance process

ID Application Process

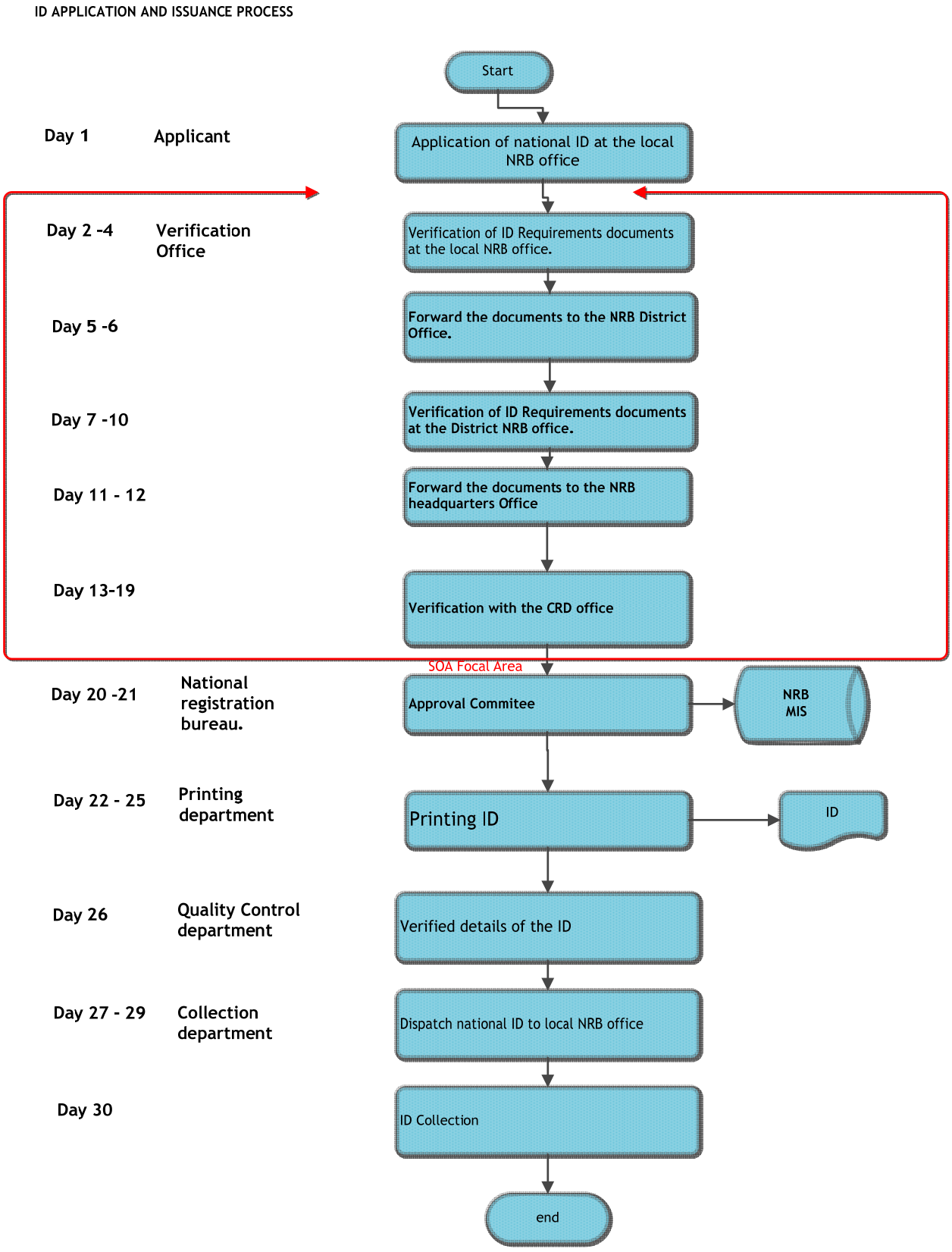


Chart 4.2 ID application/Issuance process

Birth Registration

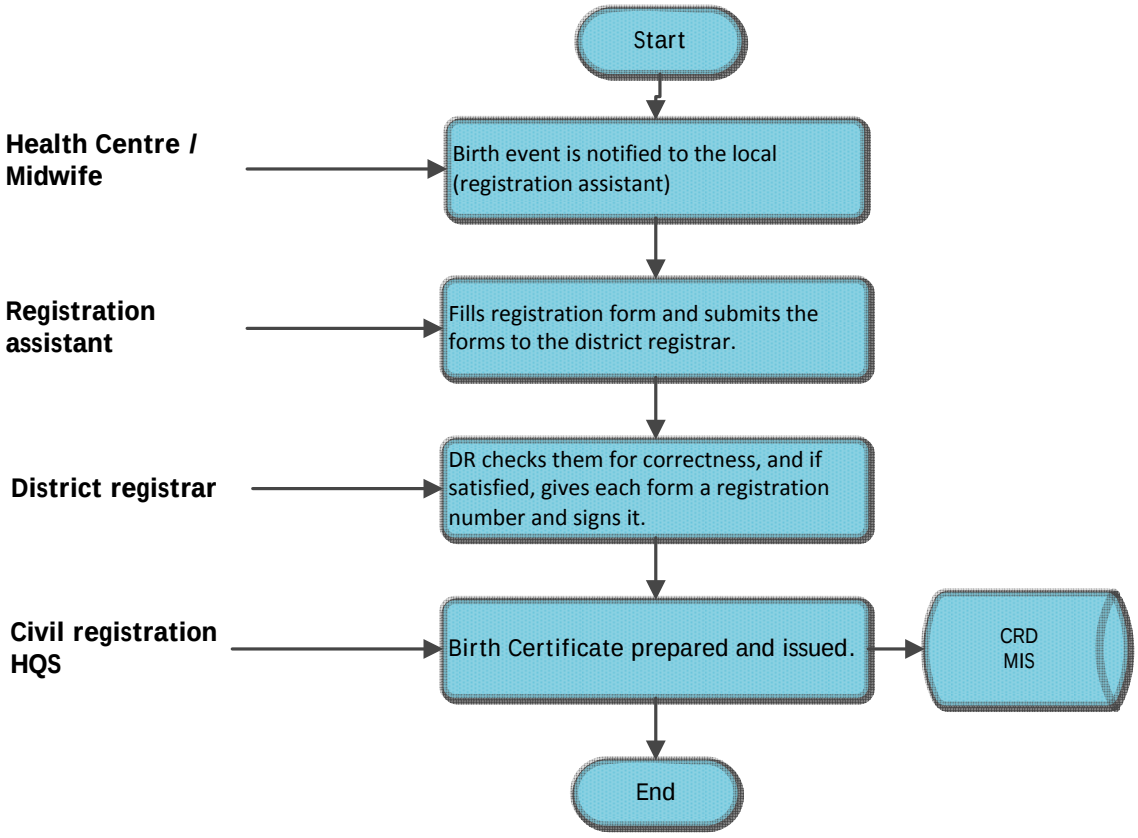


Chart 4.3 Birth Certificate application/Issuance process

These current processes pose a number of challenges to the users because they are characterized by;

i) Inefficiencies,

ii) Effort duplication and

iii) Redundant data entry

It is therefore not easy to integrate across the existing business systems. The Processing a passport for instance involves interaction with many other different systems for verification and confirmation of the applicant's details. The systems involved are often details located disparately within and outside the enterprise.

4.1.2 Solution Management

I have described a SOA based system software prototype for integrating the process of passports with two other key departments within MIRP. Service Oriented Architecture (SOA) was a response to these urgent needs by integrating or wrapping legacy applications (systems) to expose the required service functionality - This approach built a SOA enterprise solution using existing capabilities directly but exposes them through new service interfaces.

With its rich features of SOA guarantees a unique user experience, unified look and feel across applications, effortless deployment of application through "CLICK ONCE" technology shortened development to acquisition cycles.

TO realize the full potential of SOA integration solution, I have described Service Oriented Modeling Architecture (SOMA) concepts and the key activities of the method that are recommended to identify, specify, realize, implement, and deploy services at the MIRP in the subsequent phase.

4.1.3 Project Scope and Product Features

The system permits:

i) Users of Immigration Department to capture the passport registration details of applicants.

ii) Allow them to exchange the registration details remotely with the other two relevant departments; National Registration Bureau.

4.1.4 Product Perspective

The SOA system is a new system for managing communication transactions and information sharing between the two departments, replacing the current manual spreadsheet system. The component diagram below illustrates the various entities and system interfaces. The system is expected to evolve over several releases, reflecting changing user's requirements as well changes in technical requirements of other external systems.

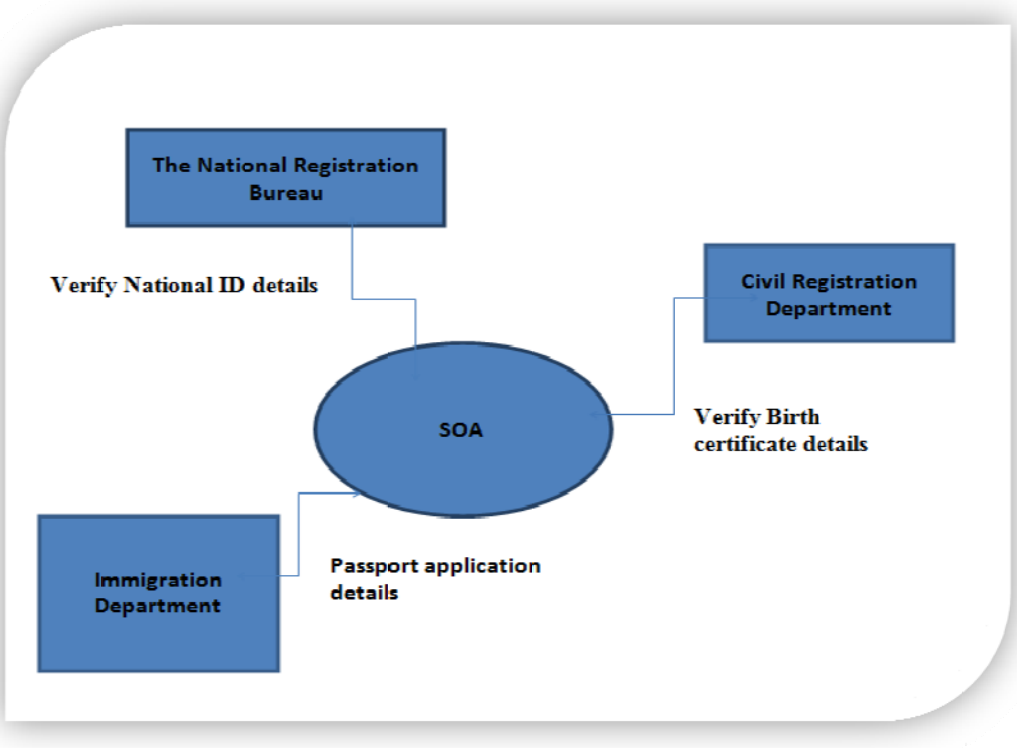


Fig. 4.1 SOA Component Diagram

4.2 IDENTIFICATION PHASE

Identification phase involved identifying of three fundamental constructs of SOA: Services, components and flows. The following identification techniques were used to identify candidate services at the MIRP before creating service portfolio.

- Goal Service Model
- Domain Decomposition

4.2.1 Goal service Model - GSM

A generalized statement of business goals at the MIRP were decomposed into sub goals.

Key Performance Indicators (KPIs) and metrics were identified and used to measure, monitor and quantify the success of SOA solution in fulfilling business needs.

In the Ministry of Immigration and Registration of Persons, the highest goals are indicated in T*able 4.1.*

Goal and Sub Goals	KPI's	Metric	Services
National Registration bureau ***Goal:*** Register all persons over 18 years old by issuing ID cards. ***Sub:*** Improving efficiency of ID processing. Sub: Ensure all persons over 18 years have registration documents. Sub: Ensure ID particulars are current at all times.	Percentage of ID's issued against target population.	Number of ID cards issued after application.	• ID Registration and issuance. • Verification of application details from CRD. • ID Replacement. • Change of ID particulars. • Check status of ID application.
Immigration Department ***Goal:*** Issue travel documents to eligible citizens. ***Sub:*** Improving efficiency of passport processing.	Percentage of Passport issued against applied passports.	Number of passports issued after application.	• Passport Registration and issuance. • Verification of application details from NRB. • Renewal of passports. • Checking status of passport application.
Goal 2: Issuance of Visas to eligible visitors.	Percentage of Visas issued against applied Visas.	Number of Visa issued after application.	• Visa application and issuance. • Check status of applied Visa. • Receive payments.
Goal 3: Issuance of work permits to eligible professionals.	Percentage of work permits issued against applied.	Number of permits issued after application.	• Work permit application and issuance. • Verification of applicant's details. • Receive payments.
Goal 4: Issuance of citizenship to eligible persons.	Percentage of citizenship issued against applied.	Number of citizenship issued after application.	• Citizenship application and issuance. • Verification and vetting of applicants details. • Receive payments.

Civil Registration Department			
Goal 1: Register all births within the country.	Percentage of all births recorded against the fatality rate.	Number of births certificates issued.	• Birth registration. • Verification of applicants ID details. • Birth certificate issuance. • Replacement of Birth certificates. • Registration of street children and orphans.
Goal 2: Register all deaths within the country. ***Sub:*** Improving efficiency of birth certificate processing.	Percentage of all deaths recorded against the mortality rate.	Number of death certificates issued.	• Death registration. • Verification ID details. • Payment receipt.

Table 4.1Goal-service model for MIRP

4.2.2 Domain Decomposition

A top-down analysis of business domains and business process modeling was used alongside the goal-service modeling to identify service components, flows.

A functional area analysis was carried out that provided a structural partitioning of the business domains at the MIRP into distinct functional area as shown in *Table 4.2*

Domains	Functional Areas	Subsystems
Immigration Department	Control of the entry and exit of persons Issuance of travel of documents to Kenyans (Passports) Regulation of the residency and employment of non-Kenyans Provision of consular services in Kenya Missions abroad	Passport Registration process Work permit processing Printing process
National Registration Bureau	Identification and registration of Kenyan citizens above the age of 18 years Production and issuance of secure Identification documents Management of a comprehensive database of all registered persons Detection and prevention of illegal registration.	ID Registration process Printing
Civil Registration	Registration of births and deaths Preservation, security and custody of births and deaths records. Issuance of births and deaths certificates Processing of vital statistics - both natality (birth statistics) and mortality (death statistics) Re-registration upon legitimating and recognition	Birth Registration process Death Registration process. Printing

Table 4.2 Functional area and subsystems within the MIRP

4.2.3 Process Decomposition

Process decomposition and process modeling were used to identify services and their service flows at the MIRP. A process hierarchy was developed as shown in the two tables below; *Table 4.3 (a) and 4.3 (b)*

Passport Processing	**Passport Processing**
Fill form	Fill in the FORM
Verify form details	Birth Certificate
Make payment	ID Card Number
Verify details with other departments	Certificate of naturalization
	Three passport size photographs
Indexing	Verify form attachments
Approval	Make Payment
Printing	Issue a token receipt to the applicant.
Issuance	Photo department where photo is captured digitally
	Verifies the details in liaison with:
	National Registration Bureau (NRB)
	Civil Registration Department (CRD).
	INDEXING
	File opening
	Investigate if close relatives have been issued with passports before, if so merge files.
	Data capture - update computer files of relatives.
	Scan FORM PP1 and the recommender's letter and save details on the computer.
	Approval
	Recommending officer- recommends the application
	Approval officer - Approves passport based on recommended application
	Allocates or issues a unique number to the approved passport
	Passport is printed.
	Checks the quality of the passport material
	Verifies every detail of the passport.
	Generates a unique bar code, scans the passport and saves details in a computer file.
	Sends the printed passport back to the issuing officer for final signature.
	Issuance
	The issuing officer hands passport to collection for dispatch.
	Applicant MUST produce the original ID and original payment receipt.

Table 4.3(a) Process decomposition: Passport processing

ID Processing	**ID Processing**
Application at the local station	Application at the local station with any of the documents below.
Verification of ID Requirements	Birth certificates
Forward the documents at the District Office.	Religious Certificates
	School leaving Certificates
Verification of ID Requirements at the District office.	Age assessment certificate from a medical officer of health.
	Child health card
Forward the documents to the NRB headquarters Office	Notification of birth
	Letter of administrative officer-Chief/Assistant Chief
Verification with other departments	Sworn Affidavits for late registration)
Approval	Proof in support of citizenship.
Printing	Parents ID card for Kenyans by birth)
Issuance	Certificate of registration as a Kenyan citizen (Kenyan by registration or naturalization)
	Verification of ID Requirements
	Forward the documents at the District Office.
	Verification of ID Requirements at the District office.
	Forward the documents to the NRB headquarters Office
	Verification with other departments
	Civil Registration Department
	Approval
	Printing
	Issuance

Table 4.3(b) Process decomposition: ID processing

4.2.4 Service portfolio

All business-aligned services that collectively support business processes and goals at the MIRP were identified and created into the following service portfolio.

Service	Description	Status	Associations	
			Function	**Goal**
ID application	Request for ID issuance at the National Registration Bureau.	**Candidate**	**National Registration Bureau**	Register all persons over 18 years old.
ID Replacement	Request for ID replacement at the National Registration Bureau.	**Candidate**	**National Registration Bureau**	Ensure all persons over 18 years have registration documents.
Change of ID details	Request for ID details change due to marriage *etc*.	**Candidate**	**National Registration Bureau**	**Ensure all persons over 18 years have up-to-date ID's**
ID Application status	Check status of ID application.	**Candidate**	**National Registration Bureau**	Register all persons over 18 years old.
Passport application	Request for passport issuance at the immigration department.	**Candidate**	**Immigration**	Facilitate foreign travel by Kenyans.
Passport renewals	Request for passport renewal upon expiry.	**Candidate**	**Immigration**	Facilitate foreign travel by Kenyans.
Passport Application status.	Check status of passport application.	**Candidate**	**Immigration**	Facilitate foreign travel by Kenyans.
Visa Issuance	Application by foreigners to enter the country.	**Candidate**	**Immigration**	Issuance of Visa's to eligible Kenyans.
Work permit application	Request for work permit issuance at the immigration department.	**Candidate**	**Immigration**	**Facilitate legal documentation to foreigners willing to work in Kenya.**
Citizenship application.	Application for citizenship by non Kenyans.	**Candidate**	**Immigration**	Issuance of citizenship to eligible persons.
Birth Certificate processing.	Request for Birth certificate issuance at the Civil Registration Department.	**Candidate**	**Civil Registration Department**	Register all births in the country.
Verification of details of Birth registration applicant	Verify parents details from the national registration bureau on request from CRD.	**Expose**	**Civil Registration Department**	Register all births in the country.
Verification of details of ID applicant.	Verify birth details from the Civil registry on request from immigration department and	**Expose**	**National Registration bureau**	Register all persons over 18 years old.

	relay results from the CRD department.		**(NRB)**	
Verification of details for passport applicant.	Verify ID details on request from immigration department and relay results from the NRB department	**Expose**	**Immigration**	Facilitate travel by Kenyans.
Printing of passport or ID	Printing of the various documents after approval.		**Immigration / National Registration Bureau**	Register all persons over 18 years old.
Payments	**Payment of relevant fees to facilitate the above services.**		**All**	

Table 4.4 MIRP Service portfolio

4.3 SPECIFICATION PHASE

Service specification is the phase where the SOA was designed. This was achieved by the high-level design of services, components and flows.

4.3.1 Service Specification

Service specification establishes and validates service exposure decisions and derivation of the high-level service model. (Velichko Ginev Sarev University of Sofia, Bulgaria: Process and realization of SOA centralized system, *Pg 38*)

Service specification as a core of services modeling activity required the provision of;

- Names of services.
- Required interfaces indicating their functional capabilities
- The dependencies of services on other services, components, applications, composition of service and the flows among services.

Service operations were invoked to execute a business function in an IT implementation. Service operations comprised of only those services that were exposed in the service portfolio.

4.3.2 Service exposure decisions

The mapping provided a potential set of exposure decisions for the SOA. Redundancy elimination was done focusing on the ability to reuse the candidate service across multiple composite scenarios where the specific function was needed.

Service	Expose			
		Business Alignment	**Redundancy elimination**	**Comments**
Verification of details of Birth registration applicant	**Y**	**Y**	Replacement of double verification from: National Registration Bureau (NRB)	Reduce the number of days it takes for verification.
Verification of details of ID applicant.	**Y**	**Y**	Verification done in one place as compared to the below stages. Verification of ID Requirements at local office. Forward the documents at the District Office. Verification of ID Requirements at the District office. Forward the documents to the NRB headquarters Office Verification with the Civil Registration Department.	Reduces the number of days it takes for verification.
Verification of details for passport applicant.	**Y**	**Y**	Replacement of double verification from: National Registration Bureau (NRB) Civil Registration Department (CRD).	Reduces the number of days it takes for verification.

Table 4.5(a) MIRP Service exposure decisions

4.3.3 Service Dependencies

The MIRP service model was further elaborated by service dependencies as indicated in the tables below.

Verification of details of ID applicant.

Functional Dependency	**Pre-condition dependency**
Verification of ID details will depend on Civil registration MIS for its functionality	Authentication service must have executed successfully before the current invocation can begin execution.
Processing dependency	**Post-condition dependency**
The service broker has to be invoked to complete the successful execution of the current service	None.

Table 4.5 (b) MIRP Service dependencies: verification of ID details from CRD MIS

Verification of details of passport applicant.

Functional Dependency	**Pre-condition dependency**
Verification of ID details will depend on National registration bureau MIS for its functionality Verification of ID details will depend on Civil Registration Department MIS for its functionality	Authentication service must have executed successfully before the current invocation can begin execution.
Processing dependency	**Post-condition dependency**
The service broker has to be invoked to complete the successful execution of the current service	None.

Table 4.5 (c) Verification of passport details from NRB and CRD MIS

The following Service Model diagram depicts the service dependency for a group of related services illustrating service consumers and service providers at the MIRP indicating the message flow between services.

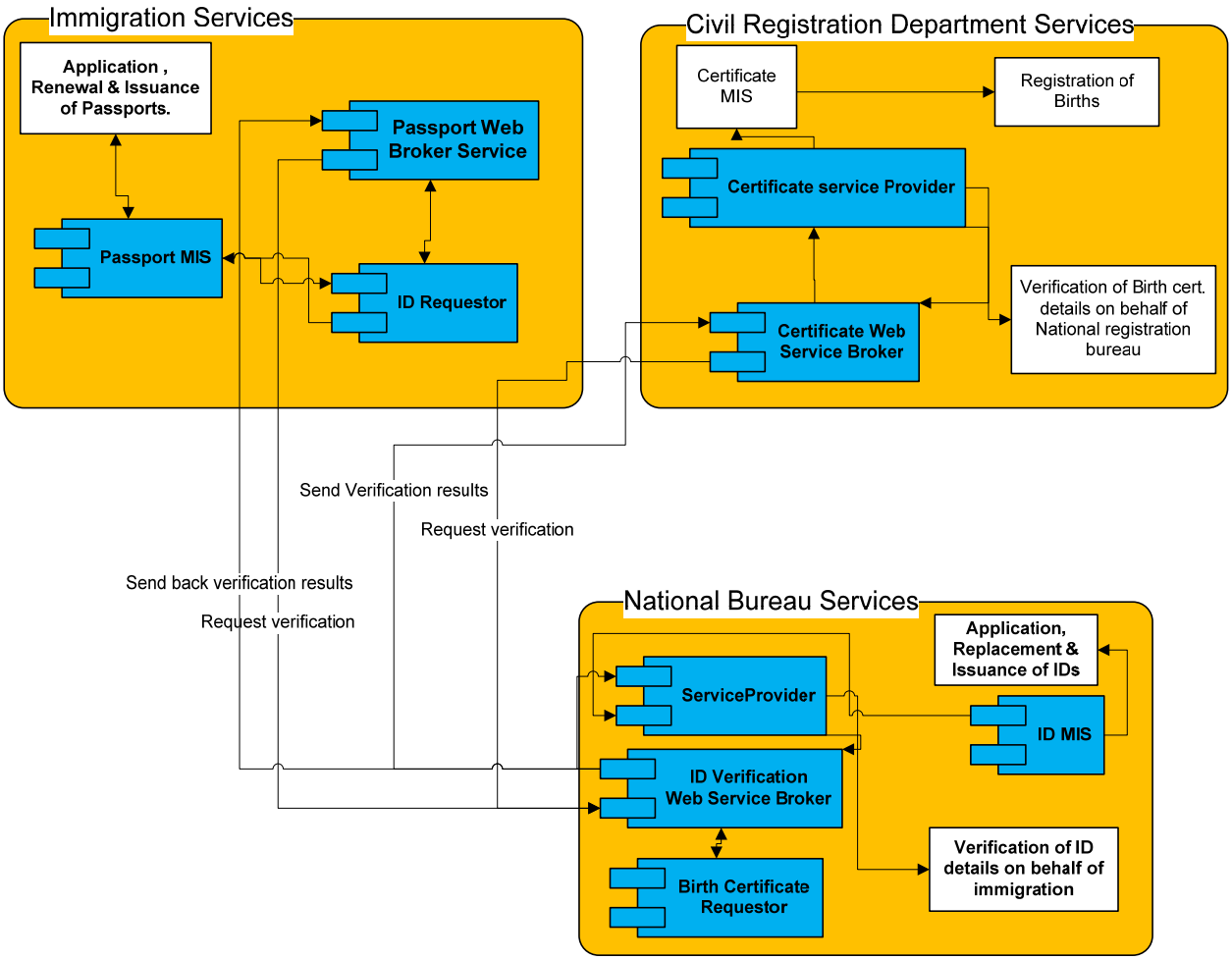


Fig. 4.2 MIRP Service model mapping

4.3.4 Service Components Description

Business Services: The tangible services provided by the business. In the case of MIRP, the business services are passport processing, ID verification and birth certificate verification

Service Providers (Functional component): A technical description of a service. The software component that fulfills a service request.

Service Brokers (Service component): A broker in the context of MIRP is a middleware application or gateway through which service requests from service providers and service requestors are made.

Service Buses: A logical channel that carries messages exchange between a service requestor and service provider.

Service Database (MIS): The data needed to fulfill a service request.

Service flows

- Every business service must be associated with at least one service database and one service provider. The MIS is the service database. The service provider is an application code /web service that services the request.
- Service providers and service requestors cannot fulfill service requests directly. They must do so through a gateway called service broker. The service broker is an application implemented as a web service that listens for both outgoing and incoming requests and determines the service endpoint to handle the request. Therefore every service requestor must bind to at least one service broker
- A service broker on one end of the communication communicates with a service broker on the other end of the tunnel. Therefore, service brokers can simultaneously be regarded as service requestors and service providers too
- A service provider queries info from the service data and performs a query based on the service request and returns the results to the service broker. The service broker at point A then routes back the result to the service broker at the end point B.

4.3.5 Subsystem analysis and Component specification

Subsystems were identified by functional decomposition of functional area as shown in *fig 5.3 (ii)* below.
During the component specification, service components were structured into a set of functional components and the technical components to support these functional components were also explored.

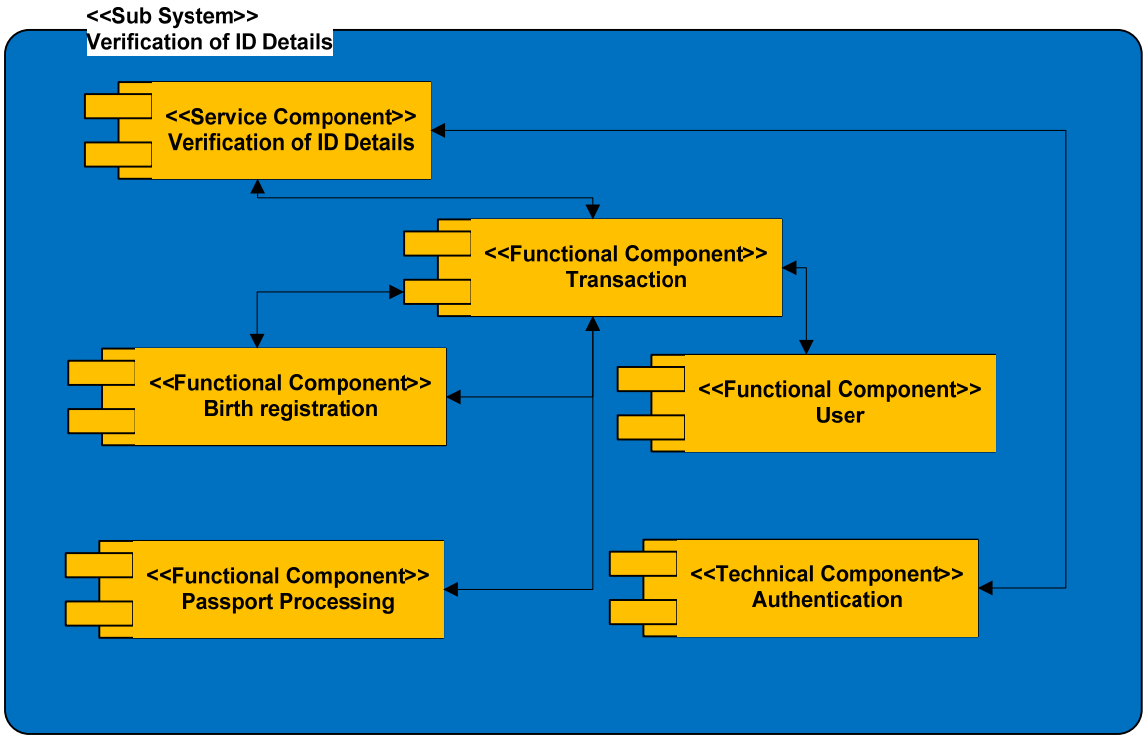


Fig. 4.3 ID Verification subsystem.

4.4 REALIZATION PHASE

Refinement of detailed components was done by populating a reference architecture that provided a snapshot view of the proposed SOA solution at the ministry of Immigration and Registration of Persons (MIRP).

MIRP SOA reference architecture:

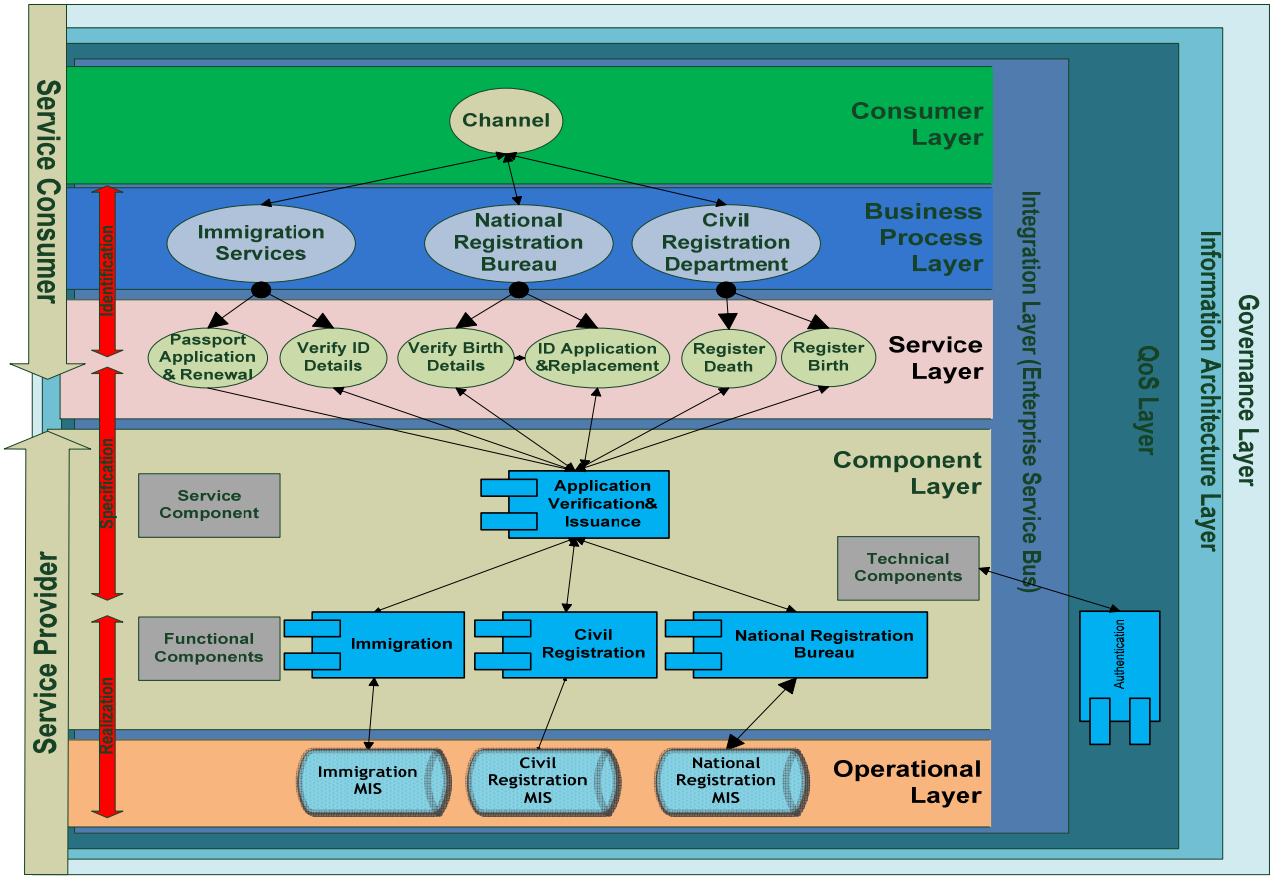


Fig. 4.4 SOA reference Architecture

4.4.1 Operational Layer

This layer has custom or packaged application assets in the application portfolio running within the MIRP. The operational layer is made up of existing application software systems; thereby used to leverage existing IT investments in implementing the SOA solution.

4.4.2 Component Layer

This layer contained software components, each of which provide the implementation for, realization of, or operation on a service. Components reflect the definition of a service, both in its functionality and its quality of service.

4.4.3 Services Layer

This layer consisted of all the services defined within MIRP's SOA identification phase. Exposed services also resided in this layer; they were discovered and invoked or possibly choreographed to create a composite service.

4.4.4 Business Process Layer

This layer showed information exchange flow between participants (individual users and business entities), resources, and processes in a variety of forms to achieve the business goal. Business logic is used to form service flows as parallel tasks or sequential tasks based on business rules, policies, and other business requirements.

The business process layer communicates with the consumer layer (also called the presentation layer) to communicate inputs and results from the various people who use the system (end users, decision makers, system administrators) through Web portals or business-to-business (B2B) programs. The key performance indicators (KPIs) for each task or process defined in the QoS and business intelligence layers.

4.4.5 Consumer Layer

The consumer layer, or the presentation layer, provided the capabilities required to deliver IT functions and data to end users to meet specific usage preferences. This layer will also provide an interface for application to application communication. The consumer layer of the SOA solution stack provides the capability to quickly create the front end of business processes and composite applications to respond to changes in user needs through channels, portals, rich clients, and other mechanisms. It enables channel-independent access to those business processes supported by various application and platforms.

4.4.6 Integration Layer

The integration layer is a key enabler for an SOA because it provided the capability to mediate, route, and transport service requests from the service requester to the correct service provider. This layer enabled the integration of services through the introduction of a reliable set of capabilities. These include modest point-to-point capabilities for tightly coupled endpoint integration as well as more intelligent routing, protocol mediation, and other transformation mechanisms often provided by an enterprise service bus (ESB).

4.4.7 Quality of Service Layer

The QoS layer provided SOA with the capabilities required to realize nonfunctional requirements (NFRs). It captured, monitored logs, and signal noncompliance with any requirements relating to the relevant service qualities associated with each SOA layer. This layer acted as an observer of the other layers by emitting signals or events when a noncompliance condition was detected or anticipated.

4.4.8 Information Architecture And Business Intelligence Layer

The information architecture and business intelligence layer ensured the inclusion of key considerations pertaining to data architecture and information architectures used as the basis for the creation of business intelligence through data marts and data warehouses. This includes metadata content, which is stored in this layer, as well as information architecture and business intelligence considerations.

4.4.9 Governance Layer

The governance layer covered all aspects of business operational life-cycle management in SOA. It provided guidance and policies for making decisions about SOA and managing all aspects of SOA solution, including capacity, performance, security, and monitoring. It enables SOA governance services to be fully integrated by emphasizing the operational life-cycle management aspect of the SOA and it is well connected with layer 7 *(Quality of Service layer).*

Layer	Layer Description	Rationale	E-Governance (MIRP Context)	SOA Component Proposed
1	Operational Layer	Legacy Systems, Business Intelligence of enterprise	Custom or packaged application assets in the application portfolio running within the MIRP. Existing application software systems; used to leverage existing IT investments in implementing the SOA solution	Service Provisioning (Service Brokerage and Service Orchestration)
2	Component Layer	Maintain Quality of Services; Organize Service Level Agreements	State Data Centers, National Data Centers, Identification of Service Providers are in the agenda	Services Composition, Loosely Coupled
3	Services Layer	Business Processes, Interfaces and Orchestration	State level Grids, Connectivity to Citizen services and Interfaces with Citizens	Service Providers (Service Composition, Aggregation, Orchestration)
4	Business Process Layer	Choreography, Business Integration	Government services and business services to converge; Government and Business process Re-engineering	Service Orchestration (Supply)
5	Consumers layer	User Interfaces		Demand for services
6	Integration Layer	Intelligent interfaces, protocol mediation	Location specific contents	Mediate, route, and transport service requests from the service requester to the correct service provider.

7	Quality of Services Layer	Monitor, Manage and maintain quality of service	e-governance standards at national government level, interoperability protocols	Service orchestration
	Information Architecture Layer	Data architecture and information architectures that can also be used as the basis for the creation of business intelligence.		Data marts, data warehouses and metadata content
	Governance Layer	Capacity, performance, security, and monitoring. Enforces QoS and make appropriate application of performance metrics, based on QoS and KPIs, policies, and SOA solution-level security-enablement guidelines.		Business rules, policies for the business process, validation rules, and input and output transformations.

Table 4.6 SOA Reference Architecture Layers

4.5 IMPLEMENTATION

System Specifications

This phase specified the minimum requirements that should be put in place for the system to perform optimally.

Operating Environment

Server Side Environment

Windows/Linux

A SOAP web service to listen for incoming requests and for dispatching requests

Client Side Environment

- An authentication layer to verify communicating entities
- A SOAP client to host the services.

System Features

Functional Requirements

Verify applicant's Details

The system provides an interface for verifying applicant's details

Hardware Interfaces

- No additional hardware interfaces have been identified. This is because the prototype will run on the normal PC internet infrastructure.

Communication Interfaces

- Internet link

Performance Requirements

- Responses to requests take no longer than 7 seconds to load on the screen after the users submits the query.
- For critical operations, the system displays confirmation messages to users within 4 seconds after the user submits information to the system.

4.7.4 Security Requirements

- Service consumers were authenticated using the security architecture described during the system testing stage.
- Physical layer and network layer security were assumed to be enforced.

4.7.5 Service Interface design

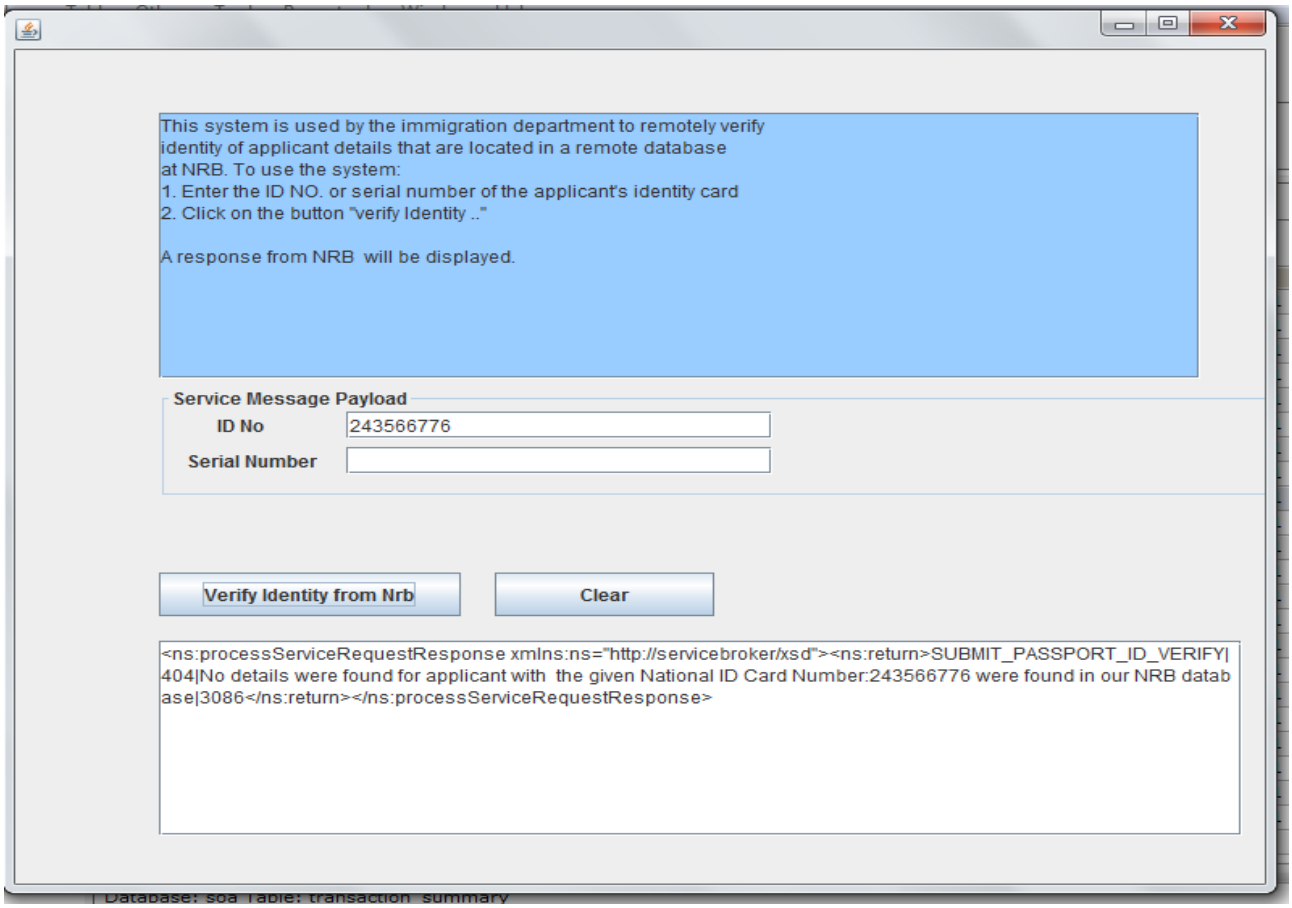


Fig.4.5 Service Interface Design

Service Deployment

In order to enforce the above architecture; the underlying components and contracts, the following tools were used:

a) The JAVA core programming language. Defined the service processors and service brokers.

b) Apache Axi2: An open source library for Java that acts as a SOAP engine. Axis2 acted as the trans-coder. It is responsible to translate Java code into SOAP/XML packets and relay it to the service container for transportation. Similarly it is responsible for decoding SOAP messages from the web-server into Java data types.

c) Tomcat 7.0 Servlet Web Container: A high performance webserver that was used to host the services.

d) MYSQL 5.0 database: Was used to host the service databases.

The implementation deployment architecture is shown in the *Fig. 4.6* below:

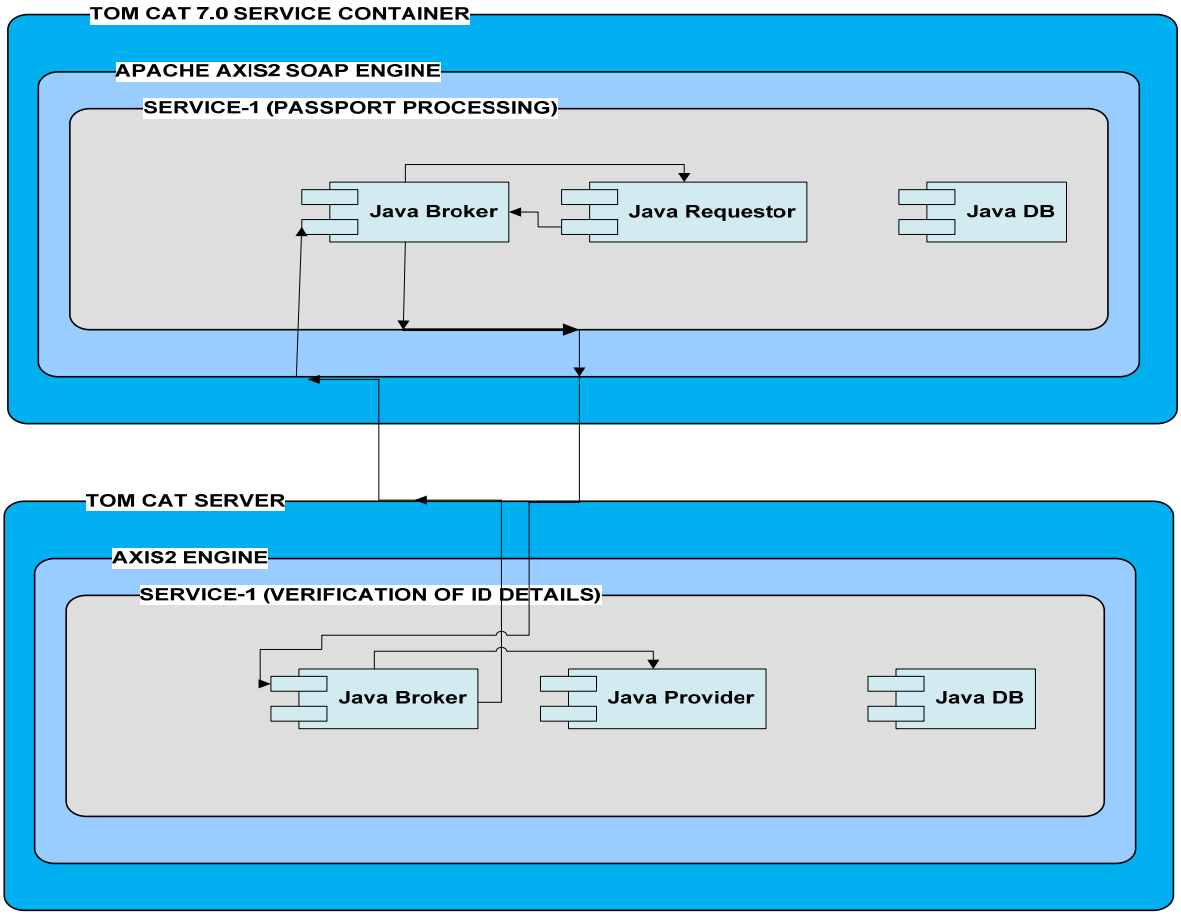


Fig. 4.6 Service Deployment Architecture

Implementation of the Service Broker: Detailed Description

Net beans ID 6.9.1 Provided support for Axis2, the SOAP engine that implemented the SOA architecture.. All our services were implemented using normal Java code then used the AXIS2 plugin to convert the code into deployable web services that communicate using SOAP. The code organization is shown in *Fig 4.7* below:

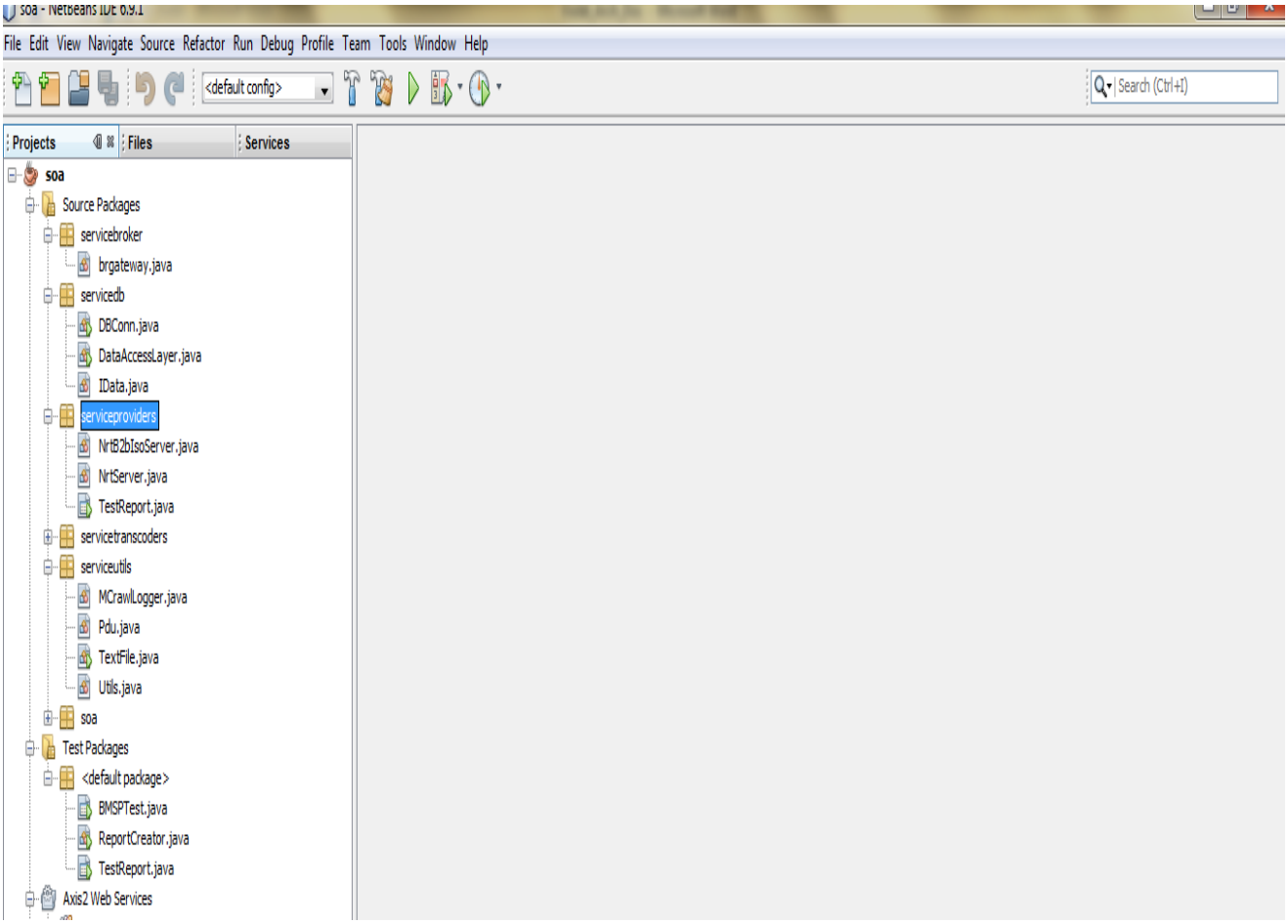


Fig. 4.7 Service Deployment: Code Structure

The code is divided into the following packages each implementing the abstract components that were identified in the architecture:

1) **ServiceDB:** A java implementation of the service data component allowing service requestors and providers to access data from the database
2) **Service Provider** Package: Implemented both the service requestor and service provider abstract components
3) **Service broker** package: Implemented the service broker interface that is exposed to the outside world
4) **Axis2 Web services:** Contains one implementation of a Java service exposed to the outside world. The Java class is called *brgateway.class* and is the concrete implementation of the service broker. The axis package allows for the conversion of java classes into SOAP packages that can then be used to communicate with a remote service broker that may be implemented in a different language.

Tomcat Webserver Management interface was used to deploy the services. Once uploaded, the services appear in the list of services.

lmrGateway

Service Description : lmrGateway

Service EPR : http://localhost:8084/b2capps/services/lmrGateway

Service Status : Active

Available Operations

- processServiceRequest

brgateway

Service Description : brgateway

Service EPR : http://localhost:8084/b2capps/services/brgateway

Service Status : Active

In the case of MIRP the service labeled “brtgateway is the SOAP service that was exposed to the outside world. The SOAP equivalent code can be viewed by double clicking the service name which produces the SOA based SOAP code: *(See appendix I)*

CHAPTER FIVE

TESTING AND RESULTS

5.1 INTRODUCTION

This chapter presents the results of the findings from the developed prototype. An estimated 1000 entries were used as sample data from the National Registration Bureau.

5.2 TESTING

The testing was divided into four segments for authentic results;

1. Service Authentication Testing
2. Communication Channel Testing
3. Service Request Testing
4. Test Response Times Testing

5.2.1 Service Authentication Testing

This involved testing whether prototype system is able to provide access to unauthorized users. The following fields were used to authenticate users in the system.

1. Service requestor ID
2. Services requestor password
3. Service partner ID

The authentication part of the system was eventually hidden from the user by coding the log in details into the java code. This made the system friendlier to the user.

The test results were populated in the tables below.

	Field Name	**Input**	**Response**
Service Authentication	**Service requestor ID**	**"Blank"**	|303|Transaction rejected. Unrecognized partner Id . Untrusted partner. Access denied mandatory parameter 'password'|
	Service Requestor password	**"Blank"**	|303|Transaction rejected. Unrecognized partner Id . Untrusted partner. Access denied mandatory parameter 'password'|
	Service Partner ID	**"Blank"**	|303|Transaction rejected. Unrecognized partner Id . Untrusted partner. Access denied mandatory parameter 'password'|

Service Authentication	**Service requestor ID**	**Immigration**	|303|Transaction rejected. Unrecognized partner Id 1008 . Untrusted partner. Access denied mandatory parameter 'password'|
	Service Requestor password	**Letmein**	|303|Transaction rejected. Unrecognized partner Id 1008 . Untrusted partner. Access denied mandatory parameter 'password'|
	Service Partner ID	**1008**	|303|Transaction rejected. Unrecognized partner Id 1008 . Untrusted partner. Access denied mandatory parameter 'password'|

Table 5.1(a) Service Authentication Testing - Invalid

The correct user log-in details were entered as shown in the table below, prompting for the communication channel verification.

Service Authentication	**Service requestor ID**	**Immigration**	|601Service request rejected due to a .A message bus error. Unrecognized service bus type => ||601Service request rejected due to a .A message bus error. Unrecognized service bus type =>
	Service Requestor password	**Letmein**	|601Service request rejected due to a .A message bus error. Unrecognized service bus type => ||601Service request rejected due to a .A message bus error. Unrecognized service bus type =>
	Service Partner ID	**1001**	|601Service request rejected due to a .A message bus error. Unrecognized service bus type => ||601Service request rejected due to a .A message bus error. Unrecognized service bus type =>

Table 5.1(b) Service Authentication Testing - Valid

5.2.2 Communication Channel Testing

The communication channels specify the remote service's address and the user authentication information for remote connections. Includes the originator (Where the request is coming from) and the destination (where the request should be sent back to). Message format is also specified.

1) Bus ID - Specifies which channel the service request is to be directed through. The channels are already defined in a database.

2) Service type - Specifies service request type. i.e. business to business or business to customer.

3) Service requestor ref no - This is a random number generated to differentiate the service requests.

4) PDU (Protocol Detector Unit) - This specifies the messaging format to be used. The format used is XML.

Communication Bus Signaling.	**Bus ID**	**1**	|603Service request rejected due to a .service type error. Unrecognized service type => ||603Service request rejected due to a .service type error. Unrecognized service type =>
	Service Type	**"Blank"**	|603Service request rejected due to a .service type error. Unrecognized service type => ||603Service request rejected due to a .service type error. Unrecognized service type =>
	Service Requestor Ref. No.	**"Blank"**	|603Service request rejected due to a .service type error. Unrecognized service type => ||603Service request rejected due to a .service type error. Unrecognized service type =>
	PDU	**"Blank"**	|603Service request rejected due to a .service type error. Unrecognized service type => ||603Service request rejected due to a .service type error. Unrecognized service type =>

Communication Bus Signaling.	**Bus ID**	**1**	|606Service request rejected due to a PDU error.Invalid PDU type => ||606Service request rejected due to a PDU error.Invalid PDU type =>
	Service Type	**5**	|606Service request rejected due to a PDU error.Invalid PDU type => ||606Service request rejected due to a PDU error.Invalid PDU type =>
	Service Requestor Ref. No.	**11**	|606Service request rejected due to a PDU error. Invalid PDU type => ||606Service request rejected due to a PDU error. Invalid PDU type =>
	PDU	**"Blank"**	|606Service request rejected due to a PDU error. Invalid PDU type => ||606Service request rejected due to a PDU error. Invalid PDU type =>

Communication Bus Signaling.	**Bus ID**	**1**	|606Service request rejected due to a PDU error. Invalid PDU type => ||606Service request rejected due to a PDU error. Invalid PDU type =>
	Service Type	**5**	|603Service request rejected due to a .service type error. Unrecognized service type => 4||603Service request rejected due to a .service type error. Unrecognized service type => 4
	Service Requestor Ref. No.	**11**	|603Service request rejected due to a .service type error. Unrecognized service type => 4||603Service request rejected due to a .service type error. Unrecognized service type => 4
	PDU	**submit_passport_id_verify**	|603Service request rejected due to a .service type error. Unrecognized service type => 4||603Service request rejected due to a .service type error. Unrecognized service type => 4

Table 5.2 Communication Channel Testing

5.2.3 Service Request Testing

This involved testing whether the system is able to respond to service requests from the users using the two unique identifying fields below were used to verify user requests. Either of the two fields can be used to verify the ID details from the NRB service provider in the following precedence.

1) ID Number
2) ID Serial Number

Field Name		Input	Response
Service Request Test 1	**[ID No.]**	Null	<ns:processServiceRequestResponsexmlns:ns="http://servicebroker/xsd"><ns:return>500Transaction could not completed. We are currently experiencing a technical problem. Please try later|1816</ns:return></ns:processServiceRequestResponse>
	[Serial Number]	Null	
Service Request Test 2	**[ID No.]**	111111	<ns:processServiceRequestResponse xmlns:ns="http://servicebroker/xsd"><ns:return>SUBMIT_PASSPORT_ID_VERIFY|404|No details were found for applicant with the given National ID Card Number:111111 were found in our NRB database|609</ns:return></ns:processServiceRequestResponse>
	[Serial Number]		
Service Request Test 3	**[ID No.]**		
	[Serial Number]	222222	<ns:processServiceRequestResponse xmlns:ns="http://servicebroker/xsd"><ns:return>SUBMIT_PASSPORT_ID_VERIFY|404|No details were found for applicant with the given National ID Card Serial Number:222222 were found in our NRB database|3480</ns:return></ns:processServiceRequestResponse>
Service Request Test 4	**[ID No.]**	20743145	<ns:processServiceRequestResponse xmlns:ns="http://servicebroker/xsd"><ns:return>SUBMIT_PASSPORT_ID_VERIFY|200{R1={IssuancePlace=Mombasa, IdSerialNumber=56566110, ParentsName=NICHOLAS KIGO KABIRU, DOB=27/09/1985, FullName=PATRICK KINYUA GITONGA, IdNumber=20743145, Location=Mombasa, Sex=M, DateIssued=24/08/2003}}</ns:return></ns:processServiceRequestResponse>
	[Serial Number]		

Service Request Test 5	**[ID No.]**		
	[Serial Number]	5656 6211	<ns:processServiceRequestResponse xmlns:ns="http://servicebroker/xsd"><ns:return>SUBMIT_PASSPORT_ID_VERIFY|200{R1={IssuancePlace=Mombasa, IdSerialNumber=56566211, ParentsName=MARGARET NGETHI THUKU, DOB=27/09/1985, FullName=LILIAN WAMBOI MWAURA, IdNumber=22087983, Location=Mombasa, Sex=M, DateIssued=24/08/2003}}</ns:return></ns:processServiceRequestResponse>
Service Request Test 6	**[ID No.]**	2074 3145	<ns:processServiceRequestResponse xmlns:ns="http://servicebroker/xsd"><ns:return>SUBMIT_PASSPORT_ID_VERIFY|200{R1={IssuancePlace=Mombasa, IdSerialNumber=56566110, ParentsName=NICHOLAS KIGO KABIRU, DOB=27/09/1985, FullName=PATRICK KINYUA GITONGA, IdNumber=20743145, Location=Mombasa, Sex=M, DateIssued=24/08/2003}}</ns:return></ns:processServiceRequestResponse>
	[Serial Number]	5656 6211	

Table 5.3 Service Request Testing

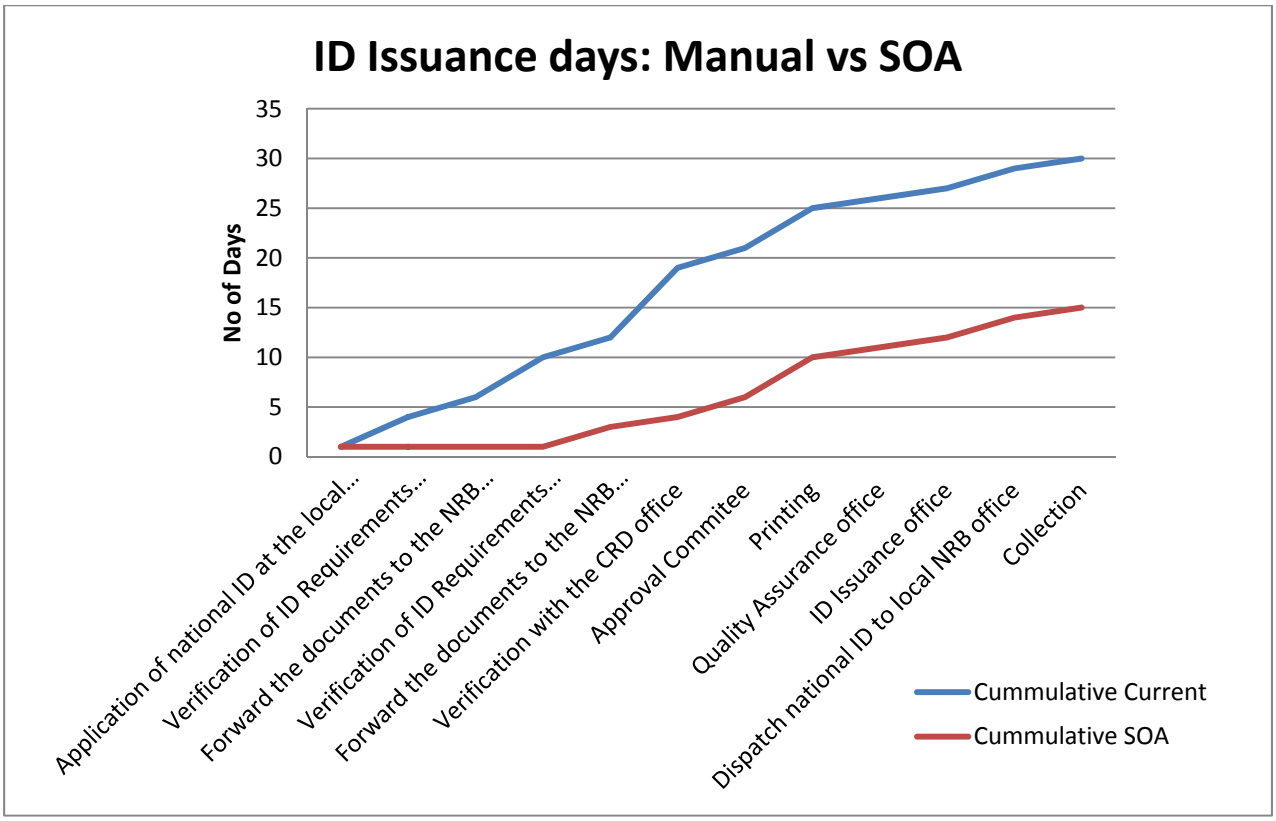
5.2.4 Test Response Times

The average response time from the service provider was tabulated below.

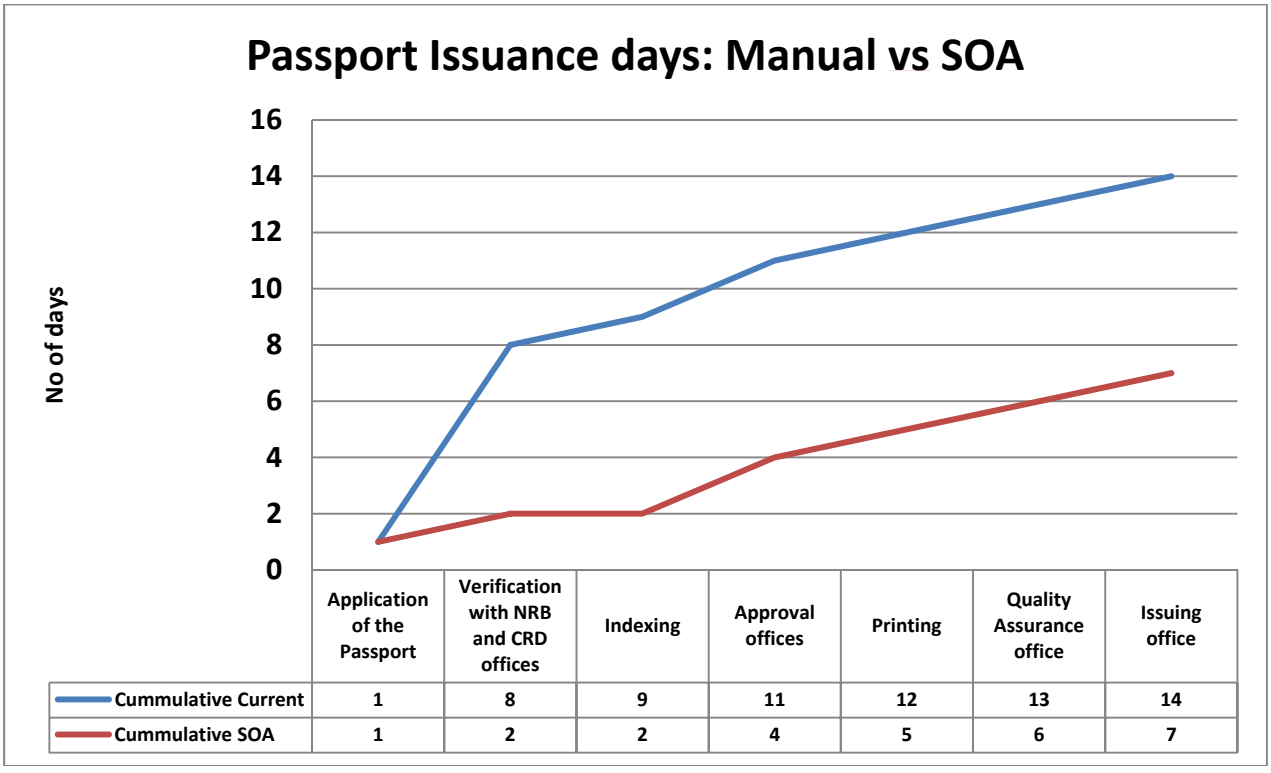
PDU	ACCOUNT_NUMBER	ACCOUNT_TYPE	START_TIME	END_TIME
submit_passport_id_verify			2012-10-21 18:49:50	2012-10-21 18:49:51
submit_passport_id_verify	111111	National ID Card Number	2012-10-21 18:52:10	2012-10-21 18:52:11
submit_passport_id_verify	222222	National ID Card Serial Number	2012-10-21 18:53:29	2012-10-21 18:53:29
submit_passport_id_verify	20743145	National ID Card Number	2012-10-21 18:55:01	2012-10-21 18:55:01
submit_passport_id_verify	56566211	National ID Card Serial Number	2012-10-21 18:56:02	2012-10-21 18:56:02
submit_passport_id_verify	20743145	National ID Card Number	2012-10-21 19:00:58	2012-10-21 19:00:58

Table 5.4 Test Response Times Testing

5.3 TESTING GRAPHS



Graph 5.1 ID Issuance Manual vs SOA



	Application of the Passport	Verification with NRB and CRD offices	Indexing	Approval offices	Printing	Quality Assurance office	Issuing office
Cummulative Current	1	8	9	11	12	13	14
Cummulative SOA	1	2	2	4	5	6	7

Graph 5.2 Passport Issuance Manual vs SOA

CHAPTER SIX

CONCLUSION AND RECOMMENDATIONS

6.1 OBJECTIVES RESULTS

Identify the business work flows, communication flows, and the common information requirements within the three departments of IMD, CRD and NRB	Identified and charted with the help of senior staff in each of the individual departments at the MIRP. These processes were then modeled and decomposed into services that are now easily shared across enterprise using the developed integration solution.
Identify the current and potential integration difficulties of the three systems and services.	Identified which include but not limited to the use of different information systems which have led to islands of information with little or no information sharing resulting in sluggish communication between the different systems and yet they are unequivocally interdependent.
Develop a SOA model of the proposed integration solution.	A SOA model of the integration solution was developed taking into consideration the disparate systems from two departments within Ministry of Immigration and Registration of Persons (MIRP). A system prototype using JAVA SOAP was then developed to implement this system ntegration model.

Table 6.1 Objectives Results

6.2 CONCLUSION

The findings from the prototype reveal that the use of SOA model that supports aggregation of information, interaction and personalization of information to specific user needs can help support contextualization and a seamless flow of information. The study focused on the Ministry of Immigration and Registration of Persons (MIRP) and can as well be replicated for other government ministries and agencies with departmental disparate systems.

The prototype study and evaluation shows that by using the SOA integration approach, the ministry will drastically cut overheads that would have been used in developing expensive ERP (Enterprise Resource Planning) systems. The prototype development has demonstrated that there is a shortened process of the application of custodial documents and as a result citizens are contended with the expedited acquisition of the same documents.

6.3 RECOMMENDATIONS

SOA as an integration solution approach has demonstrated an improved government information sharing and consequently increasing service delivery to citizens.

Further study of SOA integration platform to include a Citizen Interface that will allow external users to make applications online, query their application status, make online payments for all the services requested for. This will make it possible for the ministry to use the SOA infrastructure to link up all its other legacy systems and offer clients a one stop centre for all their services.

Further study into SOA platform should allow for the verified results to be automated and uploaded to a different system. For instance, In the case of passport application, when ID details are requested and verified from the NRB system the data should then be uploaded automatically on to the Passport registration system.

6.4 LIMITATIONS

Research time was very limited because of SOA's wide scope with projected wide application across different spectrum. Government bureaucracy made the access to processes in the various ministries a daunting uphill task.

REFERENCES

1. Arsanjani, A., "Service oriented modeling and architecture - How to identify, specify, and realize services for your SOA", (2004).
2. Arsanjani, A., "Service Oriented Architecture and Process", (2006).
3. Arsanjani, A., *et al, "IBM's Systems Journal, VOL 47: SOMA, Service-Oriented Modelling and Architecture"*, IBM Corporation, (2008).
4. Barry, D. K., Kaufman, M., "Web Services and Service Oriented Architectures", (2003).
5. Brown, A.W, Johnston, S., Kelly K., "Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications", (2002).
6. Cantara, M., "Common features of external service providers' SOA frameworks and offerings", (2005).
7. Dostal, W. *et al*, "Service-oriented Architecture meet Web Services", (2005).
8. Duermeyer, K., Holley, K., "Bridging business value to SOA: SOA best practices", (2005).
9. Erl, T. "Service-oriented architecture: A field guide to integrating XML and web services." (2004).
10. Erl, T. "Service-Oriented Architecture: Concepts, Technology, and Design", (2005).
11. Erl, T., "Service-Oriented Architecture: Concepts, Technology, and Design". (2005).
12. Erradi, A. *et al*, "SOAF: An architectural framework for service definition and realization", (2006).
13. Gustavo, A. *et al*, "Web Services, Concepts Architecture and Applications", (2004).
14. Joaquin, A., "Service-Oriented Architecture: Achieving Interoperability in Government with XML and Web Services", (2006).
15. Jones, S.C., "Toward an acceptable definition of service", (2005).
16. Mike, R. *et al*, "Applied SOA, Service-Oriented Architecture and Design Strategies." (2008).
17. Misra, Harekrishna, "Understanding SOA Perspective of e-Governance in Indian Context: Case Based Study" (2009). *BLED 2009 Proceedings.* Paper 21.
18. Papazoglou, M. P., Van den Heuvel, W. J., "Service-oriented design and development methodology", International Journal of Web Engineering and Technology (IJWET), 2006.

ONLINE RESOURCES

i) Chappell, D., Jewell, T., (2002) Java Web Services http://www.imadeyoucry.com/ebooks/Books/OReilly.Java.Web%20Services.pdf

ii) Heeks, R., (2004), "e-government for development": http://www.egov4dev.org/success/definitions.shtml#definition

iii) Kenya immigration department http://www.identity.go.ke/index.php?option=com_content&view=article&id=90&Itemid=115

iv) Liu, M., (2009) "NET BPEL Microsoft SOA Web Services". http://www.packtpub.com/article/soa-service-oriented-architecture

v) Mittal, K., *"Service Oriented Unified Process (SOUP)"*, available from http://www.kunalmittal.com/html.soup.shtml

vi) O'Brien, R., http://russellobrien.hubpages.com/hub/Integration-Architecture-Explained

vii) SOA Life cycle http://searchsoa.techtarget.com/tip/Understanding-the-SOA-lifecycle

viii) SOA-Glossary, *Cambridge Technology Enterprises"*, available from http://www.ctepl.com/soaterms.shtml.

ix) Vigne, M., (2002), E-Government: Tools of the Trade by Mark LaVigne, Center for Technology in Government/University at Albany/SUNY www.netcaucus.org/books/egov2001/pdf/e-govt.pdf

x) Zimmermann, O., *et al*, "Analysis and design techniques for Service-Oriented Development and Integration", available from http://www.perspectivesonwebservices.de/download/INF05-ServiceModelingv11.pdf.

Appendix I: SOAP CODE

<wsdl:definitions

targetNamespace="http://servicebroker/"><wsdl:documentation>brgateway</wsdl:documentation><wsdl:types><xs:schema attributeFormDefault="qualified" elementFormDefault="qualified" targetNamespace="http://servicebroker/xsd"><xs:complexType name="Exception"><xs:sequence><xs:elementminOccurs="0" name="Exception" nillable="true" type="xs:anyType"/></xs:sequence></xs:complexType><xs:element name="Exception"><xs:complexType><xs:sequence><xs:elementminOccurs="0" name="Exception" nillable="true" type="ns0:Exception"/></xs:sequence></xs:complexType></xs:element><xs:element name="processServiceRequest"><xs:complexType><xs:sequence><xs:element minOccurs="0" name="userName" nillable="true" type="xs:string"/><xs:elementminOccurs="0" name="password" nillable="true" type="xs:string"/><xs:elementminOccurs="0" name="partnerId" nillable="true" type="xs:string"/><xs:elementminOccurs="0" name="channelType" nillable="true" type="xs:string"/><xs:elementminOccurs="0" name="serviceType" nillable="true" type="xs:string"/><xs:elementminOccurs="0" name="pduType" nillable="true" type="xs:string"/><xs:elementminOccurs="0" name="serviceRequestRefNo" nillable="true" type="xs:string"/><xs:elementminOccurs="0" name="serviceRequestPayload" nillable="true" type="xs:string"/></xs:sequence></xs:complexType></xs:element><xs:element name="processServiceRequestResponse"><xs:complexType><xs:sequence><xs:element minOccurs="0" name="return" nillable="true" type="xs:string"/></xs:sequence></xs:complexType></xs:element></xs:schema></wsdl:types><wsdl:message name="processServiceRequestRequest"><wsdl:part name="parameters" element="ns0:processServiceRequest"/></wsdl:message><wsdl:message name="processServiceRequestResponse"><wsdl:part name="parameters" element="ns0:processServiceRequestResponse"/></wsdl:message><wsdl:message name="Exception"><wsdl:part name="parameters" element="ns0:Exception"/></wsdl:message><wsdl:portType name="brgatewayPortType"><wsdl:operation name="processServiceRequest"><wsdl:input message="axis2:processServiceRequestRequest" wsaw:Action="urn:processServiceRequest"/><wsdl:output message="axis2:processServiceRequestResponse" wsaw:Action="urn:processServiceRequestResponse"/><wsdl:fault message="axis2:Exception" name="Exception" wsaw:Action="urn:processServiceRequestException"/></wsdl:operation></wsdl:portType><wsdl:binding name="brgatewaySOAP11Binding" type="axis2:brgatewayPortType"><soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/><wsdl:operation name="processServiceRequest"><soap:operationsoapAction="urn:processServiceRequest" style="document"/><wsdl:input><soap:body use="literal"/></wsdl:input><wsdl:output><soap:body use="literal"/></wsdl:output><wsdl:fault name="Exception"><soap:fault use="literal"

name="Exception"/></wsdl:fault></wsdl:operation></wsdl:binding><wsdl:binding name="brgatewaySOAP12Binding" type="axis2:brgatewayPortType"><soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/><wsdl:operation name="processServiceRequest"><soap12:operation soapAction="urn:processServiceRequest" style="document"/><wsdl:input><soap12:body use="literal"/></wsdl:input><wsdl:output><soap12:body use="literal"/></wsdl:output><wsdl:fault name="Exception"><soap12:fault use="literal" name="Exception"/></wsdl:fault></wsdl:operation></wsdl:binding><wsdl:binding name="brgatewayHttpBinding" type="axis2:brgatewayPortType"><http:binding verb="POST"/><wsdl:operation name="processServiceRequest"><http:operation location="brgateway/processServiceRequest"/><wsdl:input><mime:content type="text/xml" part="parameters"/></wsdl:input><wsdl:output><mime:content type="text/xml" part="parameters"/></wsdl:output></wsdl:operation></wsdl:binding><wsdl:service name="brgateway"><wsdl:port name="brgatewaySOAP11port_http" binding="axis2:brgatewaySOAP11Binding"><soap:address location="http://localhost:8084/b2capps/services/brgateway.brgatewaySOAP11port_http/"/></wsdl:port><wsdl:port name="brgatewaySOAP12port_http" binding="axis2:brgatewaySOAP12Binding"><soap12:address location="http://localhost:8084/b2capps/services/brgateway.brgatewaySOAP12port_http/"/></wsdl:port><wsdl:port name="brgatewayHttpport" binding="axis2:brgatewayHttpBinding"><http:address location="http://localhost:8084/b2capps/services/brgateway.brgatewayHttpport/"/></wsdl:port></wsdl:service>

Appendix II: SERVICE BROKER PROGRAM CODE

```
packageservicebroker;
import  servicedb.DataAccessLayer;
import  serviceutils.MCrawlLogger;
import  serviceproviders.NrtServer;
importserviceutils.Utils;
importserviceutils.Pdu;
importjava.util.*;
/**
 *
 * @author Vmasava
 */
public class brgateway {
public String serviceResponse = null;
publicbrgateway (){
   }
public String processServiceRequest(String userName, String password,
         String partnerId, String channelType,
          String serviceType, String pduType, String serviceRequestRefNo, String serviceRequestPayload)
throws Exception{
try{
if( !isTrustedPartner(partnerId)){
serviceResponse = pduType ;
serviceResponse ="|303";
serviceResponse =   serviceResponse + "|Transaction rejected. "
                    + "Unrecognized partner Id. Untrusted partner. Access denied "
```

```
                    + "mandatory parameter 'password'";
serviceResponse =   serviceResponse + "|" + serviceRequestRefNo;
returnserviceResponse;
          }
if( !isValidPartnerCredentials(userName, password)){

serviceResponse  =pduType ;
serviceResponse ="|304";
serviceResponse =    serviceResponse + "|Service request rejected."
                   + "The partner was a trusted partner but with Invalid credentials."
                   + "Either the username or password supplied was invalid "
                    + "mandatory parameter 'password'";
serviceResponse =    serviceResponse + "|" + serviceRequestRefNo;
return  serviceResponse;
          }
if(!isValidChannel(channelType)){
serviceResponse  =pduType ;
serviceResponse = "|601";
serviceResponse =  serviceResponse +  "Service request rejected due to a ."
+ "A message bus error. Unrecognized service bus type =>" +channelType;
serviceResponse =   serviceResponse + "|"+ serviceResponse;
returnserviceResponse;
             }
if(!isValidService(serviceType)){
serviceResponse  =pduType ;
serviceResponse = "|603";
serviceResponse =  serviceResponse +  "Service request rejected due to a ."
+ "service type error. Unrecognized service type =>" +serviceType;
serviceResponse =   serviceResponse + "|"+ serviceResponse;

returnserviceResponse;
             }
if(!
doesServiceBelongToChannel(channelType,serviceType)){
serviceResponse  =pduType ;
serviceResponse = "|604";
serviceResponse =  serviceResponse +  "Service request rejected due to a ."
+ "Channel Service mapping error. Failed to associate the service type  => "
                        + serviceType + " with the channel " + channelType;
serviceResponse =   serviceResponse + "|"+ serviceResponse;
returnserviceResponse;
             }
if(!Utils.fromArrayToVector(Pdu.VALID_PDU_LIST).contains(pduType)){
                // Invalid Pdu type
serviceResponse  =pduType ;
serviceResponse = "|606";
serviceResponse =  serviceResponse +  "Service request rejected due to a  PDU error."
                        + "Invalid PDU type  => " + pduType ;
serviceResponse =   serviceResponse + "|"+ serviceResponse;
returnserviceResponse;
          }
pduType = pduType.toUpperCase();

if(pduType.equals( "submit_passport_id_verify")){

return  NrtServer.processB2cPayment
                    (channelType, serviceType, userName, partnerId,
password, pduType, serviceRequestRefNo, serviceRequestPayload);
          }
```

```
//             String address = getChannelServiceProcessor(channelType, serviceType);
//             if(address.equals("")|| address.isEmpty()||address == null){
//
//                serviceResponse = "604";
//                serviceResponse=  serviceResponse +  "Service request rejected due to a ."
//                        + "Channel Service mapping error. No service end point was found to process the "
//                        + "service request for service type  => " + serviceType + " of channel " + channelType;
//                   serviceResponse =   serviceResponse + "|"+ serviceResponse;
//
//             }
//          else{
//                 serviceResponse = "600";
//                    serviceResponse=  serviceResponse +  "| Got service processor url :" + address + " .Service
request rejected due to a ."
//                        + "Channel Service mapping error. No service end point was found to process the "
//                        + "service request for service type  => " + serviceType + " of channel " + channelType;
//                   serviceResponse =   serviceResponse + "|"+ serviceRequestRefNo;
//
//             }
}catch(Exception e){
System.out.println("Mcrawl|Found error " + e.getMessage());
System.out.println("Mcrawl|Found error " + e.getCause());
e.printStackTrace();
      }
returnserviceResponse;
   }
protected static booleanisValidChannel(String channel) throws Exception{
        Vector<String>params = new Vector<String>();
        Vector<String> fields = new Vector<String>();
fields.add("ChannelId");
params.add("1");
params.add(channel);
        String query = "SELECT ChannelId FROM servicebus WHERE ChannelStatus=? AND
ChannelId=?";
if(DataAccessLayer.selectTransaction(query, fields, params).size()>0){
return true;
}else{ return false;}

   }
protectedbooleanisValidService (String serviceType) throws Exception{
        Vector<String>params = new Vector<String>();
        Vector<String> fields = new Vector<String>();
fields.add("BusServiceId");
params.add("1");
params.add(serviceType);
        String query = "SELECT BusServiceId FROM servicetypes WHERE BusServiceStatus=? AND
BusServiceId=?";
if(DataAccessLayer.selectTransaction(query, fields, params).size()>0){
return true;
}else{ return false;}
   }
protectedbooleandoesServiceBelongToChannel(String channel, String serviceType) throws Exception{
        Vector<String>params = new Vector<String>();
        Vector<String> fields = new Vector<String>();
fields.add("ChannelId");
fields.add("BusinessServiceId");
params.add(channel);
params.add(serviceType);
```

```
            String query = "SELECT ChannelId,BusinessServiceId  FROM busservicetypes WHERE ChannelId=?  AND  BusinessServiceId=?";
if(DataAccessLayer.selectTransaction(query, fields, params).size()>0){
return true;
}else{ return false;}
   }
protected String getChannelServiceProcessor(String channel, String serviceType) throws Exception{

        Vector<String>params = new Vector<String>();
        Vector<String> fields = new Vector<String>();
fields.add("appServiceAddress");
params.add(channel);
params.add(serviceType);
        String query = "SELECT appServiceAddress FROM serviceprovidersWHERE "
             + " ChannelServiceIdIN(SELECT ChannelServiceId FROM busservicetypes WHERE ChannelId=? AND BusinessServiceId=?)";
HashMap<String,String>addressMap = DataAccessLayer.selectTransaction(query, fields, params).get("R1");
        String address = addressMap.get("appServiceAddress");
return address;
   }
protected static booleanisTrustedPartner(String partnerID) throws Exception{
booleanisTrustedPartner = false;
   String query = "SELECT partnerID from serviceauth WHERE partnerID=?";
   Vector<String>resultSet = new Vector<String>();
resultSet.add("partnerID");
            Vector<String>paramSet = new Vector<String>();
paramSet.add(partnerID);
HashMap<String,HashMap<String,String>>partnerQueryResult=
DataAccessLayer.selectTransaction(query,resultSet,paramSet);
System.out.println("Mcrawl|" + query + "Result=>" +partnerQueryResult );
if(partnerQueryResult.size()>0){
isTrustedPartner = true;
   }
returnisTrustedPartner;
}
protected static booleanisValidPartnerCredentials(String userName, String password) throws Exception{
booleanvalidCredentials  = false;
   String query = "SELECT partnerUserName,partnerPassword from serviceauth WHERE partnerUsername=? AND partnerPassword=?";
   Vector<String>resultSet = new Vector<String>();
resultSet.add("partnerUserName");
resultSet.add("partnerPassword");
            Vector<String>paramSet = new Vector<String>();
paramSet.add(userName);
paramSet.add(password);
HashMap<String,HashMap<String,String>>partnerQueryResult =
DataAccessLayer.selectTransaction(query,resultSet,paramSet);
System.out.println("Mcrawl|" + query + "Result=>" +partnerQueryResult );
if(partnerQueryResult.size()>0){
validCredentials = true;
   }
returnvalidCredentials;
   }
}
```