



**University of Fort Hare**  
*Together in Excellence*

**Design and Development of a Context Sensitive Rural Development Software  
Application for eService Provisioning**

By

**Lizo Masikisiki**

Thesis submitted to the Department of Computer Science in partial fulfillment of  
the requirements for the Degree Master of Science

**Supervised by: Mr. Ngwenya Sikhumbuzo**

**Co-Supervised by: Prof. Mamello Thinyane**

**Department of Computer Science**

**Submitted: January 2018**

## **Declaration**

I, the undersigned hereby declare that, to the best of my knowledge, the content of this dissertation is my own and has not been previously submitted to any institution. All sources of information that were used in this research have been duly acknowledged.

Full Name: Lizo Masikisiki

---

Student Number: 200903375

---

Signature:



---

## **Abstract**

After more than a decade since South Africa realized the importance of Information and Communication Technology (ICT) and the role it can play to deliver services, the country is still confronted by a number of challenges that hinder the implementation of a fully-fledged ICT-based system in a form of electronic government to better deliver services and information. While rural development remains as one of the country's greatest concerns, ICT is among approaches and perspectives that are recognized for not only accelerating rural development but also for providing the country's economic growth. This research was set to investigate approaches to implement ICT solutions for rural development and service provisioning in the context of electronic government. The research focused more on the technical skills to implement such ICT systems for the benefit of rural development and e-government stakeholders that have an interest in design and development of an integrated and interoperable solution to accelerate service delivery, especially in rural communities. A mixed methods approach was used throughout the research accompanied by an evolutionary prototyping to develop the desired prototype. A study was then conducted to gain an understanding of the state and the needs of rural communities to date.

The results of the study yielded a number of urbanized service providers that rural dwellers need to timeously visit to consume services. Scenarios to design and develop the intended prototypes were then drawn from these results. The overall implementation of the prototypes produced an integrated platform that allows multiple disparate systems to communicate, share and use information. The qualities of the prototype are what this research recommends to relevant stakeholders in order to implement an integrated and interoperable e-government system that elevates rural development programs and service delivery.

## **Acknowledgements**

I would first like to thank God for giving me the strength and hope to complete this work.

I am grateful to my Supervisor Mr. S Ngwenya and the entire Department of Computer Science at the University of Fort Hare for giving the opportunity and guidance to undertake this research.

I would like to thank National Research Foundation (NRF) for providing me with enormous financial support during the course of this research.

To my family and friends, I am grateful for the love and support that you have shown me throughout the course of this work.

Lastly, to the two most important women in my life, my grandmother I. Masikisiki and my mother N. Masikisiki, I cannot thank you enough for the genuine love, support and guidance you show me every day.

## Acronyms

CoE	Centre of Excellence
DAC	Discretionary Access Control
E-government	Electronic Government
ESB	Enterprise Service Bus
G2B	Government to Businesses
G2C	Government to Citizens
G2G	Government to Government
GWEA	Government-Wide Enterprise Architecture
ICT	Information Communication Technology
IT	Information Technology
ICT4D	Information Communication Technology for Development
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
MAC	Mandatory Access Control
MIOS	Minimum Interoperability Standards
MVC	Model View Controller
OSGI	Open Service Gateway Initiative
RBAC	Role Based Access Control
REST	Representational State Transfer
SAPS	South Africa Police Service
SARS	South African Revenue Service
SITA	State Information Technology Agency
SLL	Siyakhula Living Lab
SOAP	Simple Object Access Protocol
TSC	Thusong Service enters
XML	Extensible Markup Language
HTTP	HyperText Transfer Protocol
RMI	Remote Method Invocation
RPC	Remote Procedure Call

COBRA	Common Object Request Broker Architecture
MOM	Message-Oriented-Middleware
DCOM	Distributed Component Object Model
DCE	Distributed Computing Environment
JPA	Java Persistence API
DoH	Department of Health
STS	Spring Tool Suite
UAMS	User Account Management System
JMS	Java Messaging System
SUT	System Under Test

## Table of contents

Declaration .....	ii
Abstract .....	iii
Acknowledgements .....	iv
Acronyms .....	v
Table of contents .....	vii
List of Figures .....	xiii
List of Listings .....	xv
1 Chapter One: Introduction and Background .....	1
1.1 Introduction .....	1
1.2 Background Information .....	3
1.2.1 Information and communication technology for development (ICT4D) .....	3
1.2.2 Adoption of e-Services .....	5
1.2.3 E-judiciary Systems .....	5
1.2.4 Adoption of eLearning systems .....	5
1.3 Problem statement .....	6
1.4 Research Objective .....	7
1.5 Research Questions .....	7
1.6 Research Methodology .....	7
1.7 Expected Results .....	8
1.8 Target Audience .....	8
1.9 Thesis Structure .....	8
1.10 Conclusion .....	9
2 Chapter Two: Literature Review .....	10
2.1 Introduction .....	10

2.1.1	Government ICT House of Values .....	10
2.1.2	ICT and ICT4D.....	11
2.2	ICT4D Initiatives.....	12
2.2.1	Living Labs .....	12
2.2.1.1	Siyakhula Living Lab (SLL).....	13
2.2.1.2	Tsilitswa Village .....	13
2.2.1.3	Happy Valley (KZN) .....	13
2.2.1.4	Thusong Service Centers (TSCs).....	14
2.3	E-Government .....	14
2.3.1	Stages of e-Government Implementation .....	16
2.3.2	E-government Sectors.....	17
2.4	E-Government in Africa.....	18
2.5	E-government implementation in South Africa .....	19
2.6	Distributed Systems.....	20
2.7	System Integration.....	21
2.8	Communication Layer Integration .....	21
2.8.1	Business Logic Integration .....	21
2.8.2	Data Integration .....	22
2.8.3	Data Warehousing .....	23
2.8.4	Virtual Data Integration.....	24
2.8.5	Enterprise Application Integration .....	24
2.9	Interoperability .....	25
2.10	Web Services.....	26
2.10.1	Web Service Standards .....	26
2.10.1.1	REST Web Services .....	26



2.10.1.2	SOAP Web Services .....	27
2.11	Information Security .....	27
2.12	Access Control Models .....	28
2.13	Conclusion.....	29
3	Chapter Three: Methodology and Data Analysis .....	30
3.1	Introduction .....	30
3.2	Methods.....	30
3.2.1	Distribution of Questionnaires.....	30
3.2.2	Interviews .....	31
3.2.3	Literature review .....	31
3.2.4	Prototyping Model .....	31
3.3	Data Analysis .....	32
3.4	Data Analysis Summary.....	36
3.5	Requirement Elicitation and System Design.....	36
3.5.1	Research Scenarios .....	36
3.5.1.1	Scenario 1 – User Account Management System (UAMS) .....	36
3.5.1.2	Scenario 2 – Custom Application .....	37
3.5.1.3	Scenario 3 – Birth Certificate Application Manager .....	37
3.5.1.4	Scenario 4 – Maternity Management System .....	38
3.5.2	Non-Functional Requirement .....	39
3.6	Conclusion.....	39
4	Chapter Four: System Design and Technologies.....	41
4.1	System Modeling.....	41
4.1.1	Custom System-Use cases .....	41
4.1.2	User Account Management System (UAMS) - Use Case .....	43

4.1.3	Domain Creation - Sequence Diagram .....	44
4.1.4	Entity Creation - Sequence Diagram .....	45
4.1.5	Maternity System.....	46
4.1.6	Birth Certification Application .....	48
4.2	System Design.....	49
4.2.1	Architectural Design .....	49
4.2.2	Detailed Design .....	51
4.2.3	Spring Application.....	52
4.2.3.1	Presentation layer.....	52
4.2.3.2	Service Layer .....	53
4.2.3.3	Data Layer .....	53
4.2.3.4	Integration Layer .....	53
4.2.4	Java Enterprise Edition (Java EE) - Enterprise Java Beans (EJBs).....	53
4.3	Tools and Technologies .....	54
4.4	Conclusion.....	56
5	Chapter Five: System Implementation .....	57
5.1	Introduction .....	57
5.2	Dissemination of information.....	57
5.2.1	Domain Creation.....	57
5.2.2	Domain Entity – organizational records .....	60
5.3	System Integration.....	62
5.3.1	User Account Management System.....	62
5.3.2	Document Management System .....	65
5.3.3	Maternity Management System .....	66
5.3.4	Birth Certificate System .....	67

5.4	Web Services.....	68
5.5	Integration and Interoperability.....	73
5.6	Security Implementation .....	81
5.7	Conclusion.....	84
6	Chapter Six: System Testing .....	85
6.1	Introduction .....	85
6.2	Unit Testing.....	85
6.3	Integration Testing .....	87
6.4	REST API Testing.....	91
6.5	Conclusion.....	93
7	Chapter Seven: Result and Analysis.....	94
7.1	Introduction .....	94
7.2	ICT and E-Government in South Africa .....	94
7.3	State of Rural Communities .....	96
7.4	Prototype Implementation Results .....	96
7.4.1	System customization .....	96
7.4.2	System Integration and Interoperability .....	100
7.5	Addressing Objectives and Research Questions .....	109
7.5.1	Objectives .....	109
7.6	Conclusion.....	111
8	Chapter Eight: Conclusion and Future Work .....	112
8.1	Introduction .....	112
8.2	Conclusion.....	112
8.3	Recommendations .....	114
8.4	Future Work .....	114

8.5	Conclusion.....	115
	References.....	116
	Appendix.....	125
A.	Document management System - FTP Server connection.....	125
B.	Document management System – File Upload.....	126
C.	Polling Service Route - Maternity System to Birth Certificate System.....	127
D.	User Profile Integration Service – Endpoint Definition.....	127
E.	User Profile Integration Service – Route Definition.....	129
F.	Consent Form .....	132
G.	Questionnaire Sample .....	133
H.	Proof Reading and Editing Certificate .....	134

## List of Figures

Figure 2-1: Governmental ICT house values (Segole 2010) .....	11
Figure 3-1: Rural Dwellers in Tyhume .....	32
Figure 3-2 : Level of education in Tyhume .....	33
Figure 3-3 Education vs. Employment .....	34
Figure 3-4 Employment Rate.....	34
Figure 3-5 : Services .....	35
Figure 4-1 Custom System Use Case.....	43
Figure 4-2 Account Management System Use Case.....	44
Figure 4-3 Create Domain Sequence Diagram .....	45
Figure 4-4 Create Entity Sequence Diagram .....	46
Figure 4-5 Maternity System .....	47
Figure 4-6 Patient Registration .....	48
Figure 4-7 Birth Certificate System Use cases .....	49
Figure 4-8 Architectural Design .....	50
Figure 4-9 Detailed Design .....	51
Figure 4-10 Spring Application .....	52
Figure 4-11 EJB Application .....	54
Figure 5-1 Component Interaction.....	73
Figure 6-1 JUnit Window .....	88
Figure 7-1 LDAP Structure.....	97
Figure 7-2 Secure Login Page.....	97
Figure 7-3 Domain Creation Form .....	98
Figure 7-4 Entity Creation form .....	99
Figure 7-5 Content Update Form.....	99
Figure 7-6 Public Records Viewing.....	100
Figure 7-7 Account Creation Form.....	101
Figure 7-8 Document Upload .....	102
Figure 7-9 Browse Documents .....	102
Figure 7-10 Search User Documents .....	102
Figure 7-11 Patient Registration .....	103

Figure 7-12 Pregnancy Monthly Diagnosis .....	104
Figure 7-13 Content Polling Manager .....	105
Figure 7-14 List of Applications .....	105
Figure 7-15 Parent's/Applicant's Full Profile.....	106
Figure 7-16 Navigation Menu.....	107
Figure 7-17 User Profile Service Client Route .....	107
Figure 7-18 User Profile Service Route.....	108
Figure 7-19 BC Service and Maternity Service Routes.....	109

## List of Listings

Listing 5-1 Domain Model.....	58
Listing 5-2 Domain Controller.....	58
Listing 5-3 Domain DB Creator Service .....	59
Listing 5-4 Domain Service Repository - Spring Data JPA .....	59
Listing 5-5 Organizational Entity .....	60
Listing 5-6 Save Content .....	61
Listing 5-7 View Content Controller .....	61
Listing 5-8 Load Content - Service Method .....	62
Listing 5-9 User Account Model .....	63
Listing 5-10 Create Account Controller .....	63
Listing 5-11 Create User Account .....	64
Listing 5-12 Application Context - Context Source .....	64
Listing 5-13 Document Data Model .....	65
Listing 5-14 Uploading a Document.....	65
Listing 5-15 Patient Monthly Check-up .....	66
Listing 5-16 Maternity Process.....	67
Listing 5-17 Birth Certificate Application Operations .....	68
Listing 5-18 New Application .....	68
Listing 5-19 WIDLIFY Console - Web Service Deployment.....	69
Listing 5-20 SOAP Service - SOAP UI.....	70
Listing 5-21 Service Interface - Resource Class.....	70
Listing 5-22 REST Service Server Configuration .....	71
Listing 5-23 CXF Servlet.....	72
Listing 5-24 REST Service WADL .....	72
Listing 5-25 OSGI Bundle Definition.....	75
Listing 5-26 Endpoint Definition .....	76
Listing 5-27 Routes .....	76
Listing 5-28 Multicast Routing .....	77
Listing 5-29 Apache Camel Processor .....	78
Listing 5-30 User Profile Client Routes.....	79

Listing 5-31 Automatic Record Exchange .....	80
Listing 5-32 Preparing Message Content List for SOAP .....	81
Listing 5-33 Generating Keys and Certificates .....	82
Listing 5-34 SSL Realm .....	83
Listing 5-35 Securing URL Pattern .....	83
Listing 5-36 LDAP Context Source .....	84
Listing 5-37 Authentication Provider .....	84
Listing 6-1 Unit Test Setup .....	86
Listing 6-2 List Domain Test .....	86
Listing 6-3 JUnit Rule - Expect Exception .....	87
Listing 6-4 Integration Test Setup .....	88
Listing 6-5 Integration Test .....	89
Listing 6-6 Spring MockMvc Test Setup .....	90
Listing 6-7 Spring MockMvc - Patient Registration .....	91
Listing 6-8 Rest-Assured Test Class Setup .....	91
Listing 6-9 Basic User Profile Test .....	92
Listing 6-10 Full User Profile Test .....	93



## **Chapter One: Introduction and Background**

### **1.1 Introduction**

Rural development in South Africa is still considered as one of the greatest concerns and priorities for development (Khane and Siebörger, 2011; Ngomane, 2012). By definition, rural development is a function of improving and standardizing the lifestyles of the poor in rural communities (Moseley, 2003). It has resulted in a number of organizations (NGOs and governmental organizations) and some individuals establishing initiatives or programs that seek to address the challenges faced by people living in rural areas. Ngomani (2012) on behalf of the department of performance monitoring and evaluation indicated that in order to successfully execute rural development programs collaboration of various partners is important. Most rural development programs are thus being implemented by people with different skills and experiences and perhaps with different beliefs hence they somehow differ in terms of approach although they seek to achieve a common goal. As an example, the department of education strives to improve the quality of education, particularly in rural areas by implementing rural educational programs and encouraging communal people to participate in education thereby attaining skills and knowledge necessary to contribute to rural and sustainable development (Gardiner, 2008).

Rogerson (2014) made mention of rural development as a cross-cutting mandate that requires central coordination for maximum impact. Rogerson further indicated that on the basis of rural development's priority and importance in the country, the department of rural development and land reform was established as a means to offer central coordination. This again indicates the importance and priority of rural development in the country.

One other rich definition of rural development is that of Moseley (2003) which portrays rural development as the "sustained and sustainable process of economic, social, cultural and environmental change to enhance the long-term well-being of the whole community. As this definition reveals a number of perspectives or dimensions in which rural development can be viewed, rural development has indeed attracted a number of government bodies, NGOs, and private organizations to initiate programs that are intended to address and fulfill the needs of the people living in rural marginalized areas thereby working towards a better sustained and

sustainable community. The fact that rural development touches a number of different factors including economic, social and cultural factors it encourages people to work on rural development from different perspectives. These perspectives are reflected in the South African strategic framework for rural development for 2011-2014 (Nkwinti, 2011). These include job creation, skills development, increased access to land and spatial equity.

According to the Human Sciences Research Council (HSRC), it is the mandate and the objective of the government to ensure a continuous citizen participation in government services and bidirectional communication between governments and citizens. HSRC (2016) cited that there exists a lack of proper structures to actually elevate citizen participation and offer interactive channels for bidirectional communication between government and citizens especially those who live in rural communities. While Ngomane (2012) has indicated that in order to move rural development forward it is important to have support for collaboration and access to information, the challenge of improper communication and lower citizen participation as cited earlier is a crucial issue in this regard.

Another concern in South Africa is the issue of the urban-rural digital divide. This is based on the findings of the overall internet use in the country which is said to be very much biased towards urban areas (Conradie *et al.*, 2010). This digital divide is one of the contributing factors towards a slow rural development in the country (Scott, Thinyane and Terzoli, 2009). Information and Communication Technology (ICT) solutions are often perceived as vehicles to bridge the digital divide between rural and urban areas (Ruxwana, 2010) and hence there is a vast number of ICT4D initiatives that are being run countrywide especially in marginalized communities (Conradie, Morris and Jacobs, 2003; Scott, Thinyane and Terzoli, 2009)

It is evident from the above that rural development in its nature is a very much diverse domain in terms of perspectives in which stakeholder put together plans and devise solutions to challenges faced by rural dwellers. As such, ICT has also been adopted as an angle or an approach to improving lives of people in rural areas in the context of ICT for development (ICT4D). With a practical example of Fort Hare University and Rhodes University in an initiative known as Siyakhula Living Lab (SLL), ICT4D programs seek to act against the challenges faced by the poor in rural areas through utilization and implementation of Information and Communication

Technology (ICT) services (Khane and Siebörger, 2011). These services made it possible for people to interact with one another.

## **1.2 Background Information**

This section gives a brief but yet solid background of ICT utilization as a means to intervene in rural development and service administration and provisioning.

### **1.2.1 Information and communication technology for development (ICT4D)**

Information and communication technology has been recognized as a tool that is capable of transforming the country's economy including social and political lives of people living in rural areas (Graham 2010; Goswami 2014). It is used in this research as a means to contribute to rural development. A vast number of scholars have maintained the fact that knowledge could be key for rural development on the basis that, if people living in rural marginalized communities were knowledgeable enough, they could fight against the challenges that they are facing accordingly (Swift 2009; Khane & Siebörger 2011; Burch 2007). Having said that, ICT is also recognized as the basis of knowledge sharing and representation (Huysman and Wit, 2002).

The Literature has also revealed that a number of countries have adopted ICT in the context of electronic government as a method to deliver services and information to citizens. It could be ideal to unveil in this research countries that are known to have successfully implemented ICT and/or electronic government as it is of interest in this research. The United states and Singapore are among the top regions that witnessed a successful integration of ICT and government service provisioning (Ke and Wei, 2004; Lee, Tan and Trimi, 2005). Taking the case of Singapore which was ranked second in the Accenture's survey report three times in a row labeled as an innovative leader in the implementation of e-government services (Lee, Tan and Trimi, 2005), it could be key to understand the way in which Singapore managed to get to the top.

After the government of Singapore had realized its way to the success of the e-government implementation, a program called customization was started with the aim of maximizing the value of e-movement and agencies were urged to see themselves as one government that collaborates and shares information (Ke and Wei, 2004). It was after then that IT managers and all other agencies came together and arranged quarterly meetings where they would update one other on a

number of issues pertaining to development of e-government framework. Technologies and best practices are among other things that would be discussed in the that quarterly meeting (Ke and Wei, 2004).

The Literature also points out to the issue of interoperability between ICT systems (including e-government systems) that are meant for better services delivery. Interoperability in the sense of ICT systems can be defined as the ability of two or more ICT systems to seamlessly exchange information and use it regardless of the differences in the implementation (Fong, 2007). A number of tools, principles, frameworks and architectures exist nowadays to provide features that allow implemented of system integration. Service Oriented Architecture (SOA) is recognized as an architecture that provides interoperability, flexibility and standardization through description, recovery, and invocation of services offered by the integrated systems (Madoukh and Baraka, 2013).

The government of South Africa also believes that to accelerate and improve public service delivery there is a need for a seamless flow of information across all the levels and departments of the government (Moleki, 2007). In as far as ICT is concerned, this seamless flow of information across varying levels and departments is the interoperability of the ICT systems that are deployed within those departments and agencies (Guijarro, 2007).

A challenge of the urban-rural digital divide in South Africa has been reported by many researchers in the ICT4D field (Conradie, Morris and Jacobs, 2003; Scott, 2010). The use of internet in South Africa is still geographically biased towards the urban areas while rural areas are not only lagging behind on internet access but also on factors such as literacy, computer skills and high-income rates (Conradie, Morris and Jacobs, 2003). In response to these disparities, a number of private and public sector initiatives have been started to address the challenge of urban-rural digital divide and promote rural development (Conradie, Morris and Jacobs, 2003). Some of these initiatives focus mainly on bringing ICT infrastructure in rural areas as efforts to leverage lifestyles of people living in rural areas with those living in urban areas while promoting rural development. Below are some of the initiatives that focused on promoting the use ICT to improve livelihoods and better delivering of services.

### **1.2.2 Adoption of e-Services**

A vast number of individual projects in the ICT4D space have been undertaken to produce electronic services that aim at elevating the lifestyle of rural citizens. As a result, the adoption of electronic services such as electronic learning, electronic health system, electronic government systems and electronic judiciary system are reviewed in this work.

### **1.2.3 E-judiciary Systems**

These are electronic representations of standard, well-known judiciary system, with the support of computing technology to record and store legal judicial information while producing a repository for exchange and dissemination of such information between relevant users and authorities (Scott, Thinyane and Terzoli, 2009). An example of a successful adoption of the e-judiciary systems is that of Kenya although it has been reported to have failed in the past. The Kenya e-judiciary system website is updated every hour enabling the lawyers, magistrates, and judges to access the information as they need it (Wanjiku, 2008). South Africa is one of the countries that have developed and field-tested e-judiciary system in the Siyakhula Living Lab (SLL) for the purpose of storing and disseminating legal judicial information (Scott, 2010).

### **1.2.4 Adoption of eLearning systems**

It is believed that the traditional approach to education or training does not always satisfy the need of the new generation of teaching and learning (Zhang and Nunamaker, 2003). Zhang and Nunamaker also indicate that learning is shifting from instructor-centered to learner-centered approach that can be undertaken anywhere. This resulted in the adoption of e-learning systems that are deployable in a wide range of circumstances and focus on connecting learners to learning materials and also to one another (Ikeda, Ashley and Chan, 2006). One of the benefits of e-learning systems is the fact that it adapts well to needs of progressive education as it can be undertaken parallel with other work (Puustjärvi, 2004). This is part of the work done in the ICT sphere for the purpose of delivering quality education to the public. It is on that basis that we acknowledge these systems as they are part of the initiatives that seeks to provide better education that enables the public to gain necessary skills and be competent to promote rural development.

### **1.3 Problem statement**

Rural development interventions by various governmental departments, organizations, and agencies have been trying to facilitate communication and service delivery to rural people. Some of these interventions have been implemented through utilization of ICT and thus frameworks and models were developed to guide the design and development of ICTs for rural development. Part of what these frameworks focused on includes security and integration of ICT systems deployed in various levels and/or departments participating in rural development. ICT has made a remarkable contribution in rural development in South Africa. These contributions include rural-based ICT trainings to equip rural inhabitants to understand and use ICTs on a day to day basis. However, after so many years since ICT benefits were realized in South Africa, rural development is still considered to be one of the greatest concerns. There still exist challenges in implementing ICT solutions that integrate government service providers and citizens, especially those living in rural marginalized areas. Some of the challenges hindering a complete success of utilization of ICT in addressing challenges faced by rural communities include lack of knowledge and guidelines to implement relevant solutions using the defined frameworks to produce ICT solutions that play a role in rural development. Most ICT systems that are currently in place are not in line with the government ICT house of values which is estimated to have a potential in the seamless integration of ICT systems meant to deliver services to the citizens at a reasonable cost. This also points to poor or lack of communication among disparate applications that are adopted by varying rural development participants (service providers). Among the challenges that exist to date in the ICT4D and electronic government space, there is influence of political dynamics, lack of technical skills and failure to adhere to defined standards and frameworks amongst factors that are said to be hindering the successful implementation. This work, therefore, seeks to investigate approaches to implement ICT solutions for service administration and provisioning.

The main focus of this research is to investigate approaches to design and develop customizable and interoperable ICT systems that can add value to rural development.

## **1.4 Research Objective**

The main objective of this research is to develop a prototype that demonstrates how system customization and integration together with interoperability can help in the development of rural development and e-government platforms that enable rural communities to interact with government and/or service providers electronically.

- To understand the current ICT implementation for rural development in South Africa through literature.
- To investigate the range of needs of rural communities that can be addressed using ICTs.

## **1.5 Research Questions**

- What is the current state of ICT implementation for rural development in South Africa?
- What approaches can be used to design and develop ICT systems to better deliver services to citizens?
- What service do people in rural communities need from government and other urbanized service providers?
- What are the challenges hindering a successful use of ICT systems to deliver services to citizens?

## **1.6 Research Methodology**

The nature of the research itself is somewhat complex and hence combining various methods could offer flexibility to understand various domains of the research context while heading to the intended results. The research includes application development which itself has a range of methodologies and approaches that are different from those that are used in pure researches that do not involve system or application development. Literature was reviewed as means of gathering information needed as the research was undertaken. Questionnaires and interviews were used as means of gathering requirements and feedback from citizens.

Evolutionary prototyping was used as a method to develop the intended applications. It was chosen from other prototyping methods (i.e. incremental and throw-away prototyping) on the basis

that it allows the creation of the system prototype using the overall scope of the system enabling system components to be integrated during development while also putting users at the centre of the development. Evolutionary prototyping allows the developer to better and fully understand the system requirements by demonstrating and elaborating requirements as the prototype is iterated and refined (Zhang *et al.*, 2010).

## **1.7 Expected Results**

- A prototype of an interoperable platform that allows querying by South African citizens for service delivery and dissemination of information.
- A documentation of the current state of ICT implementation for rural development in South Africa

## **1.8 Target Audience**

This work seeks to contribute to the ICT4D and e-government knowledge body as some of the perspectives of development. The target audience includes but is not limited to public sector rural development and e-government stakeholders that have an interest in improving technical skills needed to design and develop ICT solution to better deliver services.

## **1.9 Thesis Structure**

This chapter has presented the main objectives of this research including the research questions. The organization of this thesis is as follows: Chapter two presents an extensive literature review as a method to get an understanding of the research and what other scholars have done. Research methodology, data collection and analysis are presented in chapter three followed by system modeling and design in chapter four. The implementation of the intended prototypes is presented in chapter five. Testing of all systems and components is presented in chapter six followed by results in chapter seven. Chapter eight concludes this work by presenting the overall conclusion and possible future extensions.



## **1.10 Conclusion**

Herein we proposed a development of an integrated rural development application that can be used to facilitate communication and service delivery for South Africa citizens, especially those who live in rural communities. The problem statement has been specified and it is based on the state of the models and frameworks together with ICT systems produced from them to better serve the public. The issue of interoperability between ICT systems meant for service delivery also forms part of the problem statement. The development of a customizable interoperable rural development application to enable rural communities to interact with government and/or service providers electronically at minimal costs is specified to be the main objective of this research. Both mixed methods approach and evolutionary prototyping (for developing the application) are proposed methodologies for this research project.

## **Chapter Two: Literature Review**

### **2.1 Introduction**

This chapter presents a literature survey on the implementation of ICT systems in the context of ICT4D and electronic government. The success stories and challenges of ICT implementation and electronic government systems in other developing countries are also reviewed in this chapter. As this work focuses more on the technical aspect of ICT implementation in this context, it is, therefore, important to also review different technologies, tools and principles that are appropriate for the context of this work.

#### **2.1.1 Government ICT House of Values**

Before we go deep into reviewing literature on the ICT and e-government implementation South Africa, it is ideal to first touch basis on some standards or frameworks that exist to date as means guide and regulate ICT implementation in South Africa in as far as e-government is concerned. Minimum Interoperability Standards or Minimum Information Interoperability Standard (MIOS) is the program is responsible for setting out the technical principles and standards for interoperability and information system coherence across the public sector. MIOS is an integral part of the Government-Wide Enterprise Architecture framework (GWEA) which was created to provide methods as well as guidelines on how enterprise architecture should be documented applied within all levels of departments and agencies connected to the government (TOGAF, 2014).

The office of government chief information officer, the government of information technology officer's council and State Information Technology Agency (SITA) constituents are responsible for leading the government-wide ICT plans and programs. These programs were previously reported to have used a variety of frameworks and methods to develop ICT plans. It was also revealed that the programs were not aligned with the government ICT house of values which part of it includes security, interoperability and reduced redundancy (Segole, 2010). It was then that a government-wide enterprise architecture (GWEA) was created to address those issues. The GWEA itself does not provide sufficient guidelines on how to implement the defined house of values (Segole, 2010).

The chief information officer (Segole, 2010) highlighted some of the challenges that the presidential commission on the transformation of government is facing. These challenges include the lack of coordination, incompatibility of systems and architectures and information technology not being business driven. In the same report, the chief information officer also shared the government ICT house of values as depicted in figure 2-1 below.

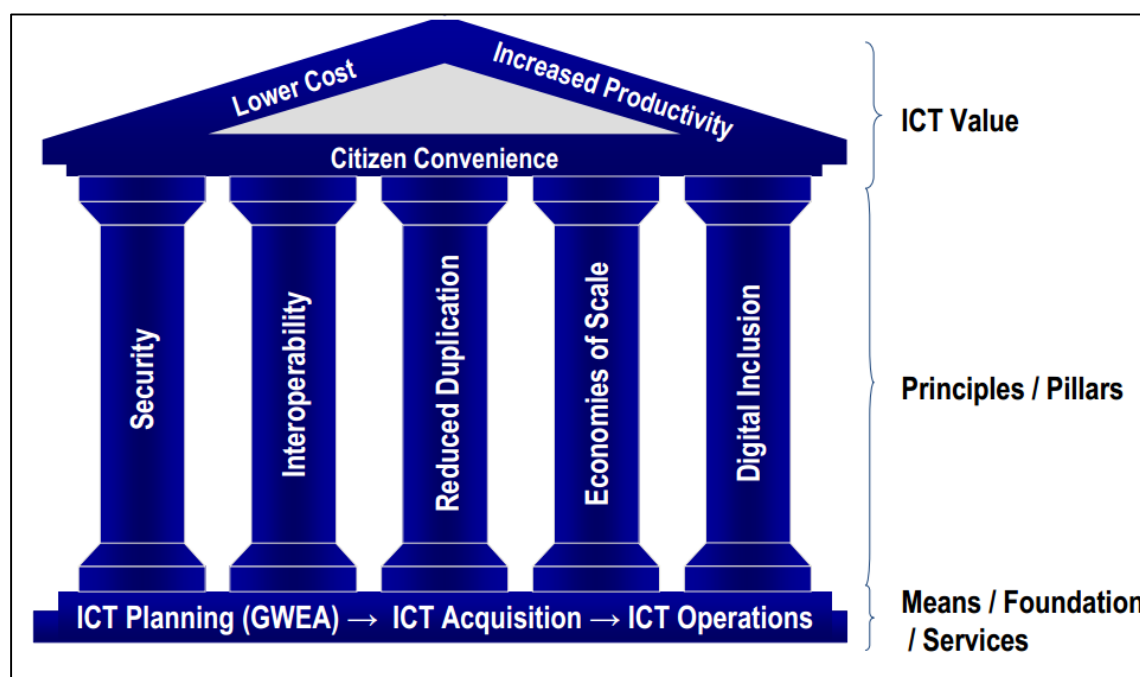


Figure 2-1: Governmental ICT house values (Segole 2010)

It is also mentioned that government developed this ICT house of values with the aim of reducing costs for government, improving efficiency, effectiveness and accessibility of government services to the citizens. The values, principles and services depicted are capable of providing security, reducing redundancy and providing other important features required in service delivery (SITA, 2014).

### 2.1.2 ICT and ICT4D

The literature on the use of ICT services has shown evidence that ICT is applicable in a wide range of domains such as agriculture and education and thus can improve service delivery. Information and Communication Technology for Development (ICT4D) is a successor of information and Communication Technology (ICT) which gained widespread adoption in the early 2000s after its potential impact on development was first recognized (Kleine and Unwin, 2009). Some ICT4D

implementations are such that information technology and community-based initiatives focus on providing access to modern information and communication technologies and train the inhabitants to use the available ICT infrastructure for their purposes (ICT4D.at). The adoption of ICT4D in South Africa has resulted in a number of projects and programs which are reviewed in the following sections.

As an attempt to absorb the advantages and benefits of the internet to extend the channels of services delivery, the public sector has made major ICT investments under the umbrella of electronic government (Kaisara and Pather, 2011). E-Government can be defined as a digital platform that enables interaction between other government departments, business and public citizens for the purpose of service delivery. The adoption of e-government is anticipated to bring a great amount of change in the way in which governments operate. Traditionally, governments relied more on meetings, print media, radio and television to communicate important information to their citizens (Mphidi, 2008). Mphidi (2008) further indicates that e-government involves new ways or approaches of doing things such as accessing education, listening to citizens and delivering both information and services.

## **2.2 ICT4D Initiatives**

### **2.2.1 Living Labs**

The idea of living labs came into existence during the late 1990s as an approach to innovation and ICT4D to provide an environment that can be referred to as an ecosystem that allows users to actively engage with the innovations and development of ICT-related products and services (Coetzee, Toit and Herselman, 2012). Gumbo and Thinyane (2012) describe living labs as initiatives that focus on a partnership with numerous stakeholders in varying levels of research, innovations, and development to create a mass impact and deal with societal challenges. Living labs have also been created for a variety of ICT-related topics from e-commerce, healthcare, transport, tourism, agriculture and governance to understand the human behavioral response to ICT and IT which can then assist in discovering new uses and service opportunities for ICT, the evaluation of new ICT solutions with users and lastly, field testing of ICT products (Gumbo and Thinyane, 2012).

### **2.2.1.1 Siyakhula Living Lab (SLL)**

Siyakhula Living Lab (SLL) was established by the Telkom Center of Excellence (CoE) hosted in the Computer Science department at the University of Fort Hare and Rhodes University with a primary goal of developing and field-testing of distributed, multifunctional community communication platform that is deployable in marginalized areas of South Africa (Gumbo and Thinyane, 2012). SLL is located at Dwesa-Cweba rural community situated in the wild coast of the former homelands of the Transkei in the Eastern Cape region of South Africa (Scott, Thinyane and Terzoli, 2009). The site was, for a long time, used by the ethnographic researchers in the Anthropology department of the Rhodes University because of its potential in economic and social development and thus it was chosen in 2005 for ICT4D project that is now known as Siyakhula Living Lab by the aforementioned CoEs (Dalvit, Siebörger and Thinyane, 2012).

### **2.2.1.2 Tsilitwa Village**

The Tsilitwa project is led by the Council for Scientific and Industrial Research (CSIR) and involves other bodies such as the department of Arts and Culture, Science and Technology (DACST), National Research Foundation (NRF) for support and funding to develop and use ICTs for the purpose of rural development (Conradie, Morris and Jacobs, 2003). Community network was implemented in the Tsilitwa village linking several sites to each other and other surrounding villages using wireless Ethernet. This network allowed CSIR to develop a telemedicine system that enables telephone calls between the clinic and hospital in Tsilitwa region for queries and instructions between nurses and doctors using voice over internet protocol (VoIP) (Chetty, Tucker and Blake, 2003).

### **2.2.1.3 Happy Valley (KZN)**

Happy Valley is the name of a rural marginalized community in KwaZulu-Natal province of South Africa. The community is typified by strong Zulu and African Traditions and still upholds ancient cultural practices just as much as the community is still governed by Zulu King and traditional leadership (Krauss, 2012).

Krauss (2012) further reveals that the community is subjected to a lot of challenges such as limited access electricity and running water, limited access to fixed-line and mobile connectivity, high rate of HIV/AIDS infection and tuberculosis.

In response to the circumstances faced by the community, a number of community development initiatives emerged with the caregivers (e.g. teachers, nurses) being the primary informants and project partners (Krauss, 2012). The ICT4D project in the happy valley community began as one of the development initiatives in late 2008 led by department of informatics, University of Pretoria (Martin Turpin 2010). Part of the aims of the project is to empower the caregivers of the happy valley community through ICT so as to empower the community entirely and thus, in partnership with UNESCO, basic IT training was offered to the Happy Valley School and Hospice (local clinic) (Krauss, 2012).

#### **2.2.1.4 Thusong Service Centers (TSCs)**

This is a public sector initiative established in the late 1990s as an approach to take information and services to South African citizens living rural marginalized communities. The plan was to produce multi-purpose integrated service centers that contribute to rural and sustainable development (Burton, 2011; GCIS, 2017). Some of the main services that TSCs were designed to offer include government services, office services, education services, local and economic development (LED) and information and communication services with the objective of providing cost effective, integrated and efficient channel for information and service delivery (GCIS, 2017).

### **2.3 E-Government**

E-government is often viewed by many as the much needed innovative change in service delivery by governments to citizens. However, literature reveals that most implementation of e-government in developing countries have not been successful due to that they do not fulfill the predefined or stated goals and objectives of the implementation (Danish, 2006).

A study on the development of e-government conducted by the United Nations (UN) identified the levels of e-government implementation in 190 nations. The levels included the integration of e-government systems and the transaction levels. The study revealed that none of the nations archived integration level and only 17 had archived transaction stage (Basu, 2004). Basu

(2004) has also indicated that the feasibility of the successful implementation of e-government is directly dependent on the governments' willingness, ability and readiness to spend on the necessary information technology and related costs.

While other nations have challenges, there are notable cases of success in some. For instance United States of America (USA), Australia and Singapore are some of the countries that have witnessed the benefits and success of e-government implementation (Backus, 2001). In contrast to fewer stories of e-government success, although there is no official statistics on the failure rate of e-government implementation, the number of e-government projects that fail worldwide is estimated to be high (Gant, 2008).

One of the success stories of e-government implementation is that of e-Europe. According to (Wauters, 2006), e-Europe initiative started as a plan to develop information society among national, regional and local governments of Europe (EU) in 1990. The plan was to develop this information society within economic and social environment thereafter use Information and Communication Technologies to perform the intended activities (Wauters, 2006). It then received an enormous commitment from the EU leaders to construct and ensure the development of knowledge-based economy and society for the benefits of public citizens. Consequently, EU is able to track, monitor and evaluate the action plan of its e-government implementation and that distinguishes it from other global countries (Basu, 2004).

Another region that recognized ICT and e-government as an approach to better deliver services to public citizens is Singapore. After the government of Singapore had realized its way to the success of the e-government implementation, a program called customization was started with the aim of maximizing the value of e-movement and agencies were urged to see themselves as one government that collaborates and shares information (Ke and Wei, 2004). It was after then that IT managers and all other agencies came together and arranged quarterly meetings where they would update one other on a number of issues pertaining the development of e-government framework including best practice and technologies together with recommended workplaces thereby trying to integrate the different applications of e-government (Ke and Wei, 2004).

### **2.3.1 Stages of e-Government Implementation**

The literature on e-government reveals that it can be implemented in varying stages of development that outline its progress and structural transformation (Layne and Lee, 2001; Seifert, 2003; Dayanidhi, 2005). The number of these stages is not fixed, as some researchers combine some of the stages while others divide them. However, hereunder these stages are looked upon as described by other scholars (Layne and Lee, 2001; Dayanidhi, 2005; Lee, Tan and Trimi, 2005) as four stages of e-government that present a model that can assist in categorizing e-government project.

➤ **Cataloguing**

This stage is all about presenting the government information in a web format thereby allowing public citizens to have access to information online (Dayanidhi, 2005). As the government website is developed it is under great pressure and interests of the media, ICT knowledgeable employees, citizens and also other contributing stakeholders (Layne and Lee, 2001). While the website is very simple and does not have many functionalities, it, however, receives a continuously increasing number of users. Varying departments of government use this one website to share information with citizens. The websites are then bound to have a number of challenges and issues that include maintenance and privacy (Layne and Lee, 2001; Seifert, 2003).

➤ **Transaction**

This is actually a progression of the initial stage after citizens and officials have agreed on taking a step further and use the advantage of the internet as a service channel. The stage-one website is then upgraded to support new functionalities that allow public citizens and businesses to complete transactions online thereby reducing paperwork and travel costs and the website is not only used for information dissemination (Layne and Lee, 2001; Seifert, 2003).

➤ **Vertical Integration**

This stage deals with the transformation of government services where scattered systems at varying government levels are integrated, establishing a link between local systems and high-level systems (Layne and Lee, 2001; Lee, Tan and Trimi, 2005; Andersen and



Henriksen, 2006). According to (Layne and Lee, 2001), this stage yields a number of sensitive challenges and issues as e-government implementation progresses to higher levels. Such challenges and issues include signal authentication, data format compatibility and indistinguishable boundaries between system levels.

➤ **Horizontal Integration**

As e-government implementation progresses to higher levels, e-government system transformation reaches the horizontal integration stage. All functions of e-government systems are in this stage integrated thereby producing a real one-stop service delivery (Layne and Lee, 2001; Lee, Tan and Trimi, 2005; Andersen and Henriksen, 2006). Databases of all functions of an e-government system are linked and thus they are able to communicate with each other and share information (Layne and Lee, 2001).

### **2.3.2 E-government Sectors**

The fact that e-government has a wide range of activities and actors which can also be referred to as stakeholders has resulted into a categorization of government service delivery into sectors (Seifert, 2003; Basu, 2004; Gant, 2008) which shall be reviewed hereunder. These sectors include Government-to-Citizens (G2C), Government-to-Businesses (G2B), and Government-to-Government (G2G). There also exists a sector that is known as Government-to-Employees (G2E), however, due to its nature being intra-agency and comprises services that fall under G2C it is sometimes disregarded as a distinct sector of e-government and put as a subset of G2C (Seifert, 2003).

➤ **Government to Citizens (G2C)**

This is referred to as a citizen-centric e-government since it focuses on provisioning and publishing of government services to public citizens online (Gant, 2008). This involves interaction between government and public citizens and electronic dissemination of information through government websites that have content that is organized according to citizen needs while also enabling citizens to make electronic transactions (Dayanidhi, 2005; Gant, 2008).

➤ **Government to Businesses (G2B)**

This form of e-government deals with strategies that use ICT to facilitate interaction between government and private sector (businesses) to procure goods and services while also managing transactions from businesses (Dayanidhi, 2005; Gant, 2008). According to Gant, e-procurement is an example of the initiatives that fall under this category of electronic government. Improved procurement practice and increased competition in this form of e-government enables sales of government goods and holds a potential for cost reduction (Seifert, 2003; Dayanidhi, 2005).

➤ **Government to Government (G2B)**

It is suggested by Seifert (2003) that, for an electronic transaction with citizens and businesses to be successful governments must first improve and update their internal systems and procedures. This form of e-government focuses on data sharing and electronic exchange of information across governments and various levels and departments of governments (Seifert, 2003; Dayanidhi, 2005; Gant, 2008).

## **2.4 E-Government in Africa**

According to (Maumbe, Owei and Alexander, 2008), e-government implementation in the context of Africa is still in its early stages and that maps to the challenges and constraints that the continent at large is faced with, including human capital, inequitable access to information, inadequate legal frameworks and cultural constraints which are the hindering factors in government implementations (Maumbe, Owei and Alexander, 2008). Nevertheless, there is still hope that as Africa embarks and invests on e-government initiatives, a number of unique challenges will be addressed. It is further stated by (Maumbe, Owei and Alexander, 2008) that, in order for African countries to successfully implement e-government, there should be adherence to the four categories of e-government implementation such as: cataloging, transactions, vertical integration and horizontal integration and serious consideration of social, cultural and economic differences is crucial (Maumbe, Owei and Alexander, 2008).

## **2.5 E-government implementation in South Africa**

According to (Kaisara and Pather, 2011), a meeting was held by the Ministry of Public Service and Administration in March 2010 where an agreement was reached to a proposal to develop an e-government platform that is both transverse and transactional. In the very same year, the Minister of Communications stated that government can use ICT to improve its efficiency and highlighted a need for the South African government to urgently adopt ICT to modernize services and improve administration (Kaisara and Pather, 2011). The government of South Africa then adopted e-government as a way of making its services accessible to public citizens in most of its dimensions (Matavire and Chigona, 2010; Kaisara and Pather, 2011). Some of the key players in the implementation and deployment of any form of ICT utilization in public sectors appear on (Cloete, 2012) Include Government Information Technology Officers Council (GITOC), State Information Technology Agency (SITA) and the Department of Public Services and Administration (DPSA). GITOC consists of Government Information Officers (GIO) from all departments and was established to monitor and coordinate public sector Information Technology (IT) initiatives thereafter give reports to the Minister of Public Service and Administration (MPSA) while GIOs acts as the secretariats of GITOC (Cloete, 2012)

South Africa to date points to a number of initiatives that have become a success in terms of electronic access to services and those include initiatives such e-filing for South African Revenue Services (SARS), e-Natis for the Department of Transport and the Western Cape provincial initiative (Matavire and Chigona, 2010; Kaisara and Pather, 2011). The initiatives that South Africa has undertaken to harness ICT with service delivery and its focus and investment on ICT made it be recognized in past years as a leading country that implements e-government in Africa although the pace of its progress is still questionable (Maumbe, Owei and Alexander, 2008).

According to what has been reviews in this chapter up to this point, it is evident that South Africa has made a great deal of investment in the ICT utilization in the public sector. The works towards full utilization of ICTs in the public sector range from rural-based ICT initiatives which of them have were presented earlier, ICT implementation in government offices and departments to setting up structures or committees to manage and monitor the implementation and its progress. However, the question remains, have these initiatives reached out to all citizens of South Africa. If they have, that would mean the known challenge of computer illiteracy and digital divide has been addressed

in South Africa. It would also imply that citizens, especially those who live in rural communities, are now able to freely use ICTs to consume services and connect with service providers digitally. The literature, however, has shown that rural parts of the country are still faced by digital divide while struggling to interact with service providers on a day to day basis.

One other question that might arise at this point is that, as a decade has gone past since e-government was initiated by the South African government, has the country managed to implement a fully-fledged e-government platform that matches the stages and sectors highlighted above? One of the ways to figure that out is by relating the stages or levels of e-government learnt from the sections above. This would then indicate how far the e-government implementation has gone in South Africa. Tons of information about government services is available on the internet, in a number of websites including DPSA and <http://www.gov.co.za/>. This alone fulfills the requirement of the first stage of e-government, cataloging. Services such as SARS e-filing system, home affairs e-channel and other services available at <http://www.eservices.gov.za/> as mentioned in the above section make it possible for South African citizens to make transactions online without having to physically visit government offices. It is then safe at this point to say that something has been done to implement the second stage of e-government, the transactional stage. What has not been clear on the e-government implementation of South Africa is the implementation of the last two stages of e-government as they are concerned about different types and levels of system integration in the context of e-government implementation. These are the same stages where interoperability as cited in the Government ICT House Values would be implemented.

## **2.6 Distributed Systems**

The rapid growth of the internet and distributed object-based technologies has triggered software industries to see the need for adopting loosely coupled architecture for web based services. These services have underlying nonfunctional requirements such as flexibility, interoperability, customizability and conformance with business processes that are also part of fundamentals for a successful integration and deployment of services in the web-based platform (Patil, Kshirsagar and Jaypal, 2014). Industries have become distributed through organizational and geographical boundaries and they adopt distributed enterprise applications which can be defined as enterprise

applications that consist of multiple modules across the networks, hence the need for integration of distributed application arises (Medjahed *et al.*, 2003; He and Xu, 2012).

## **2.7 System Integration**

Integration of distributed systems can be performed at varying levels to be discussed later in this sections and involves real-time data exchange across different system modules (He and Xu, 2012; Pascal, 2014). However, the literature reveals that there still exists some challenges associated with the general integration of enterprise systems integration such as compatibility of environments (heterogeneous) and inconsistent format of data to be shared across modules of an enterprise application (Kosanke, Vernadat and Zelm, 2014; Pascal, 2014). Below are some of the parts or levels of integration that can be performed on distributed enterprise systems as outlined by He and Xu (2012).

## **2.8 Communication Layer Integration**

Integration of distributed enterprise system requires that different modules of the enterprise system communicate for the purpose of sharing information (He and Xu, 2012). This is the primary step for any applications to interact and it is achievable by defining the set of protocols (such as HTTP) needed for transporting data irrespective of the semantics and syntax (Benatallah and Nezhad, 2008; Patil, Kshirsagar and Jaypal, 2014).

### **2.8.1 Business Logic Integration**

Integration of business logic focuses on integrating stand-alone applications with a predefined set of business protocols, standards and other values that the business consider as vital (He and Xu, 2012). Integration at this layer can broadly be divided into sub-layers that include business basic coordination, functional interfaces, policies and non-functional properties as described by Benatallah and Nezhad (Benatallah and Nezhad, 2008). To simplify business logic integration, industrial enterprises have adopted the use of middleware technologies (He and Xu, 2012). Middleware-based integration offers generic interfaces through which heterogeneous integrated systems are able to communicate while also providing functions for transforming, mapping and conversion of data shared among applications (Patil, Kshirsagar and Jaypal, 2014). Numerous

types of middleware technologies (some of which are discussed below) exists to date to facilitate the generic integration of distributed applications and specifically, business logic integration (He and Xu, 2012; Patil, Kshirsagar and Jaypal, 2014).

He and Li have highlighted some of the commonly used middleware technologies as being RPC-based middleware and Message Oriented Middleware (MOM) technologies such as distributed computing environment (DCE), distributed component object model (DCOM), Common Object Request Broker Architecture (COBRA), Java Remote Method Invocation (RMI) (Benatallah and Nezhad, 2008; He and Xu, 2012). RPC-based approach to integration enables calling operations on remote sites as they provide mechanisms for communication level integration (Medjahed *et al.*, 2003) while on the other hand message oriented middleware technologies provide messaging capabilities between disparate enterprise application components and is based on asynchronous interaction model (Curry, 2005).

### **2.8.2 Data Integration**

Access to the web information has become a habit and a commodity to a number of distinguishable bodies such government agencies, statistical specialists, energy information administration and a lot of other bodies which have a mandate to make information publicly available through information systems (Jr. and Elmagarmid, 2002; McIver and Elmagarmid, 2002). The advancement of science and technology together with information systems call for integration of data that reside on heterogeneous sources that reflect a wide range of diversity (McIver and Elmagarmid, 2002; Brazhnik and Jones, 2007; Doan, Halevy and Ives, 2012). The primary goal of data integration is to seamlessly join data among different information systems and,/or data sources that form part of distributed applications thereby allowing real-time exchange of data (Torun, 2002; Doan, Halevy and Ives, 2012; He and Xu, 2012; Pascal, 2014). This enables integrated components to seamlessly understand data such as documents and messages shared among them (Benatallah and Nezhad, 2008). According to Pascal, the key problem of general system integration lies in data integration (Pascal, 2014). Interoperability issues arise in this integration layer due to heterogeneity (e.g. semantics, structure, and syntax) of content across applications, however, these issues can be suppressed by defining mechanisms for transformation, mapping and conversion of data to be exchanged (Benatallah and Nezhad, 2008). Data integration has gained a worldwide recognition in the research community and has also gained commercial success due to

advantages that it offers which include an ability to share data across multiple data sources (Dong, Halevy and Yu, 2007), however challenges associated with heterogeneity and another factor still exists. Consequently, incompatibility issues limit the usefulness and usability of systems (Jr. and Elmagarmid, 2002). A number of technologies, frameworks, and standards such as reference model, semantic web and ontologies emerged to provide solutions to some of the known challenges of data integration (Brazhnik and Jones, 2007).

Literature reveals that data integration yields a number of architectures and approaches that can be broadly categorized into data warehousing and virtual data integration (Doan, Halevy and Ives, 2012). Torun recognizes data integration as one of three approaches, namely: scheme integration, software development and data integration itself, that can be used for database integration which implies uniform and transparent access to data from multiple sources (Torun, 2002). Doan suggests that in order archive integration data from multiple sources, semantic heterogeneity has to be bridged as it has proven to be the major challenge in data integration (Doan, Halevy and Ives, 2012). Below we review some of these concepts to gather more understanding of their purposes, approaches, and advantages they offer.

### **2.8.3 Data Warehousing**

According to Wyllie and Davies, data warehousing is a process whereby information is shared among the organization, network or nation (Wyllie and Davies, 2015). To reveal what it offers, Ponniah defines data warehousing as an informational environment that provides an integrated view of the enterprise which allows for decision support transactions while also providing a flexible and interactive source of strategic information (Ponniah, 2011).

The bigger picture depicted by Ponniah (2011) in his book makes it clear that the idea of data warehousing in its early formation was to provide an environment that produces strategic and decision support information and in the process of doing so, it encompasses data integration across various disparate data sources (Ponniah, 2011). The setup in this case is such that data from these disparate individual data sources is loaded into a physical database called warehouse (Doan, Halevy and Ives, 2012).

#### **2.8.4 Virtual Data Integration**

Virtual integration is an alternative to warehousing approach for integrating multiple data sources. Data in this approach remain the original disparate data source and access during query time (Doan, Halevy and Ives, 2012). As it might sound obvious, this approach to data integration is virtual in the sense that data stays in the sources, however clients wanting to access that data feel as if they interact with a single database (Bertossi and Bravo, 2005). Virtual data integration consists of mediators that are responsible for accepting client supplied queries fed into the global schema (GS) and thereafter collect data which is then translated through mapping by various underlying wrappers which are attached to disparate local data sources (Langeegger, Wöß and Blöchl, 2008; Hafez, Ali and Hegazy, 2013). The role of mediator in virtual integration is to provide a database-like interface that has a GS or mediated schema that has names and attributes of relations (virtual tables) and then present that interface to clients for querying (Bertossi and Bravo, 2005). Global as view (GaS) and local as view (LaV) are two main methods that can be used to define the mapping between the mediated schema and the local schemas (LS) (Bertossi and Bravo, 2005; Calì and Calvanese, 2013; Hafez, Ali and Hegazy, 2013).

What is still known to be a challenge in the web data integration is the heterogeneity of syntax, schema, and semantics across multiple data sources (Xiao, Cruz and Hsu, 2004). However, the existences of technologies such XML and semantic web technologies mitigate these challenges and offer opportunities to integrate data across multiple sources (Xiao, Cruz and Hsu, 2004; Cruz and Xiao, 2005). Consequently, as part of semantic data integration, ontologies are known to provide mechanisms to integrate heterogeneous data sources (Xiao, Cruz and Hsu, 2004). In this regard, data can be represented in an RDF (resource description framework) form which provides specifications of semantic data in a standardized and interoperable manner (Xiao, Cruz and Hsu, 2004; Cruz and Xiao, 2005).

#### **2.8.5 Enterprise Application Integration**

Over the years industries have acquired a number of disparate systems or applications that allow them to meet their business demands and objectives (Frantz and Corchuelo, 2012; He and Xu, 2014). As industries grow and applications become more distributed over networks and geographic boundaries, needs to integrate these various systems and applications have arisen as means to



effectively satisfy business needs and requirements (Kolb, 2008; He and Xu, 2014). Normally these systems and application are hosted in different business /organizational networks that implement various protocols and have some elements of heterogeneity (Frantz and Corchuelo, 2012; He and Xu, 2014). Enterprise application integration came into existence as mechanisms that offer a combination of processes, technologies, and standards that allow enterprises and industries to seamlessly integrate two or more systems thereby making them operate as one. One important aspect of EAI is the fact that systems that are involved in integration remain independent as they should be built such that they can evolve with affecting other systems. This can be referred to as loose coupling (Kolb, 2008; He and Xu, 2014).

Gregor Hohpe and Booby Woolf have provided and described in a book titled “Enterprise Integration Pattern”, a set of patterns and process that can be used to guide implementation enterprise application integration while escaping a number of known issues pertaining to EAI (Frantz and Corchuelo, 2012; Pierce, 2014). In (Pierce, 2014), it is further stated that these patterns are implemented by a number of integration frameworks that also sought to provide a model for interaction and communication among various systems components in service oriented architecture manner. Some of the most popular integration frameworks that are known to have a strong focus on these enterprise integration patterns while providing SOA-based integration solution are Apache camel and Spring Integration (Kolb, 2008; Frantz and Corchuelo, 2012; Emmersberger and Springer, 2013; Pierce, 2014; Frantz, 2015). Apache Camel is an open source, java based integration framework that provides a fluent API to design and implement enterprise integration patterns. Apache camel is mostly used through java or scalar based domain specific language although (DSL) it provides support for spring bean configurations (Frantz, 2015)

## **2.9 Interoperability**

Interoperability of information system can be defined as an ability of two or more disparate systems or components to exchange and use data stored across them (Gatautis, Kulvietis and Vitkauskaite, 2009; Loukis and Charalabidis, 2013). Its establishment across multiple bodies such as customers, suppliers, business partners has a potential to offer significant value (Loukis and Charalabidis, 2013). It has been widely adopted as a concept or a principle in the implementation of e-government initiatives. The establishment of interoperability frameworks has also assisted in pinpointing guidelines for implementation of the principle in various domains (Guijarro, 2007).

According to Curry (2012), some of the known barriers of interoperability include conceptual barriers which are all about the syntax and semantics of the exchanged data across information systems and also technological barriers which are all about the incompatibility of technologies such as communication protocols and platforms. As means to suppress these barriers, options to implement interoperability include integrated approaches where common formats for models are defined and agreed upon by concerned parties. Another option is a unified approach where common formats are defined at meta-level to enable semantics when mapping various models. Lastly, there is a federal approach which requires parties to share ontologies (Curry, 2012; Chen, 2013).

## **2.10 Web Services**

One of the most popular and appealing approaches in building software systems that are able to interact with one another from distinct locations and physical devices is web service approach (Kalin, 2009). A more precise and rich definition of web services is that of W3C as specified in (Alonso *et al.*, 2004) which portrays web services as software applications that are recognized by URIs that have interfaces and bindings with mechanisms that allow them to be defined, described and discovered. Moreover, web services have the ability to interact with other software agents by means of messages transmitted over internet-based protocols (Alonso *et al.*, 2004). They provide a lightweight but flexible mechanisms to integrate disparate software systems to make intended functionalities readily accessible (Kalin, 2009).

### **2.10.1 Web Service Standards**

The widely known categories of web services standards are SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) web services (zur Muehlen, Nickerson and Swenson, 2005). Hereunder we review the differences between REST and SOAP web services.

#### **2.10.1.1 REST Web Services**

REST web services were developed as an architectural style for building large-scale distributed systems based on URIs, UI (Uniform Interface), and self-descriptive messages and stateful interaction through hyperlinks as four driving principles (Pautasso, Zimmermann and Leymann,

2008). Part of the strengths of REST is its simplicity in terms of implementation since it leverages known standards (e.g. HTTP, XML) (Pautasso, Zimmermann and Leymann, 2008).

#### **2.10.1.2 SOAP Web Services**

SOAP is a lightweight platform independent, transport independent and operating system independent protocol for decentralized or distributed environments that make use internet and XML to exchanges information between nodes (Suda, 2003). SOAP web services expose its loosely coupled interfaces by means of XML and consumers do not have to know the underlying implementation details while services can be invoked via a self-explanatory interface that describes available methods together with their parameters and returns values (Goncalves, 2013).

### **2.11 Information Security**

The evolution of telecommunications has resulted into modern computers that are bundled with very much sophisticated software and hardware that enable the production of extremely powerful computing systems. This has encouraged the continued development of web-enabled applications that provide electronic services that are meant to be publicly available over the internet (Olden, 2002). Technology nowadays has become a commodity and an integral part of the society due to a number of advantages it offers which include access to information and productivity in the daily activities (Andress, 2014). Web applications and services are some of the technologies that have gained worldwide and enormous recognition and adoption in the last few decades owing to the fact that they are computing paradigms that provide flexible and dynamic means for people to communicate, access information, socialize and perform financial transactions over the internet (Thankachan, Ramakrishnan and Kalaiaarasi, 2014; Rafique and Humayun, 2015). However, as technology advances and becomes rapidly adopted, it also carries along security issues (Andress, 2014; Rafique and Humayun, 2015). Security in this regard means protection of information, information systems, and assets from unauthorized access which may result in damages, theft, disclosure, misuse and any other undesirable action (Andress, 2014).

As organizations become aware of the role of IT, they are also subjected to the implementation of security strategies to deal with security issues that may arise from the organization's information and its related technologies. Such security strategies are to be implemented by establishing a

comprehensive framework that improves information security program and complies with the overall organization's strategic plans (Ahmad, Maynard and Park, 2014). The fundamentals of implementing information security require one to understand and adhere to security triad principles which are confidentiality, integrity, and availability (CIA) although it can be extended to Parkarian hexed by adding control, authenticity, and utility on top of the classic CIA triad (Andress, 2014). These fundamental principles form a somewhat distinguished model of information security to which security mechanisms can be based on while also keeping the focus of security program (Andress, 2014; Farooq *et al.*, 2015).

## **2.12 Access Control Models**

Discussed below is the method of managing and controlling access to distributed applications while also exploring vital features of information security. Security in this section focuses to both authentication and authorization of users against information where authentication is generally the assurance that the users are really who they claim to be and authorization is the assurance that users are authorized to do what they request to do (Johner *et al.*, 1998; Thomas *et al.*, 2006).

Joshi and Aref (2001) have also made mention of two classes of services that are crucial for a secure internet infrastructure as access control services and communication security service. A number of varying security models for web-based application exist in information security space. These include traditional access control models such as those referred to as Discretionary Access Control (DAC) and Mandatory Access Control (MAC) models and modern models being Role-Based Access Control (RBAC) (Joshi and Aref, 2001; Li *et al.*, 2012). Various services that are known to be useful in implementing access control in distributed systems such as LDAP and capability-based Kerberos exists nowadays (Hu, Ferraiolo and Kuhn, 2006).

Discretionary Access Control (DAC) gives a certain amount of access control to the discretion of another owner. Due to its flexibility, it has been widely used by commercial sector although it reflects some loopholes and vulnerabilities (Hu, Ferraiolo and Kuhn, 2006).

Unlike DAC, MAC gives access control decision to the discretion of the central authority, not to the authorized individual (Hu, Ferraiolo and Kuhn, 2006).

In Role-Based Access Control (RBAC), access is granted to an individual based on the roles that are assigned to the particular individual by an organization to which an individual belongs (Hu, Ferraiolo and Kuhn, 2006).

## **2.13 Conclusion**

This chapter presented literature review as a way to get an understanding of the research field while investigating how different tools, technologies, standards, and principles can be used to fulfill the purpose of this research. The chapter started off with the review of ICT initiatives in the context of rural development in South Africa followed by the e-government. This provided fundamentals of this work as the research itself falls in the ICT for development and e-government space. Since the research is mainly concerned with the technical side of things in the defined context, it was necessary to review technologies and principles/concepts such distributed system, web services, system integration and interoperability. These can be very important to understand when dealing with the implementation of ICT systems in the context of ICT4D and e-government. Information security was also reviewed in this chapter as one the crucial aspects in the implementation of ICT4D and e-government systems. The review of the above areas afforded this research enough knowledge to drive towards the anticipated results.

## **Chapter Three: Methodology and Data Analysis**

### **3.1 Introduction**

While this research strives to investigate the current state of rural development in South Africa and give an insight of what still needs to be done in terms of the needs and requirements for rural inhabitants, it also strives to find an approach to design and implement of ICT systems to better deliver services to the citizens. It is in the interest of the researchers to capture every detail that relates to any of the two sides or aspects of the research and avoid any limitations that may be presented by tools, technologies, and methods. Under that basis, this research employs a mixed methods approach as its research methodology.

The research is mainly guided by extensive literature review as it provided base knowledge in a number of areas such as, ICT4D initiatives, current state and challenges of e-government in South Africa, security and a few others. However, putting citizens at the center of this research and get up-to-date information about the needs of rural dwellers is also a primary goal of this research. In that case, quantitative methods are embed as means to interact with rural dwellers and get the needed information. This, therefore, suggests that at a high level, an embedded design of mixed methods approach was adopted.

In order to understand the state of rural development in South Africa, it is important to understand what other concerned scholars and researchers have found and presented previously. This, therefore, calls for an extensive literature review. Understanding how rural inhabitants find the delivery of the existing services and what more services they need from all concerned service providers could be key. Both the distribution of questionnaires and interviews can be useful to address this requirement and hence are adopted.

### **3.2 Methods**

#### **3.2.1 Distribution of Questionnaires**

The questionnaires were distributed to three (3) rural communities as a way of inviting communities to participate in the process of elicitation of requirements to build service delivery platforms that could be useful in their lives. The participants were made aware of what the research

was all about and what was expected from them. These questionnaires were designed to encourage participants to partake freely and express their views and opinions regarding the study. The goal was to get a complete understanding of the services that citizens mostly consume or require from their urbanized service provider's offices.

### **3.2.2 Interviews**

In cases where some of the participants were not comfortable reading and filling out the questionnaires on their own, interviews were used. Questionnaires could present some limitations when other facts that interest the researcher arise during the course of the survey while interviews can be great platforms for such development (Adams and Cox, 2008). Interviews were also adopted and prepared in this research to allow a good platform to follow up on the interesting views and opinions that participants presented.

### **3.2.3 Literature review**

A literature review was perceived in this research as a convenient approach to get an overview of what has been done by the concerned stakeholders to address challenges of rural development and what still needs to be done. It is used intensively in this research to understand how ICTs solutions have been developed and implemented in rural marginalized communities of South Africa. As this research also aimed at producing a prototype demonstrating the development of ICT systems meant for rural development or public service delivery, literature review also played an important role in understanding a wide range of tools, technologies, standards and principles that are available.

### **3.2.4 Prototyping Model**

A prototype can be defined as an initial component that provides a concrete representation of an interactive design used for exploring requirement and design concepts (Bansler and Havn, 2010). Evolutionary prototyping as an iterative prototyping model was employed in this research to allow flexible iteration while moving towards the desired platform in as far as this research is concerned. Since this kind prototyping allows the prototype to evolve into the final desired product through a number of iteration with a great deal of feedback and validation (Kamlesh and Ahmad, 2008), it was then recognized as a relevant approach in this research as it has a potential to provide a

complete view and flexible iterations of the desired product. The manner in which this was followed in this research is that, with an idea of the intended platform, requirements were gathered and analyzed, the design of the overall prototype including example scenarios was then conducted followed by the actual implementation of the prototype with minor internal iterations to keep in line with the initial requirement before the first iteration undergoes evaluation and validation.

### 3.3 Data Analysis

The sections above have given an insight into the methods and techniques used to gather information about rural inhabitants and delivery of services in their communities. This section presents the analysis of data collected from the Tyhume region of Alice in the Eastern Cape Province of South Africa. This area was chosen among other rural areas that are served by Inkonkobe municipality in Alice. Although the interest of this research is not specifically in the Tyhume region, but rather in rural communities in general, factors such as Tyhume being one of the remote areas served by this rural town and its historical background made it a relevant and accessible area to conduct the study. Three locations were then chosen in this area to get at least 50 participants giving details of their situations and needs. The figure below shows gender and age group distribution for people living these communities.

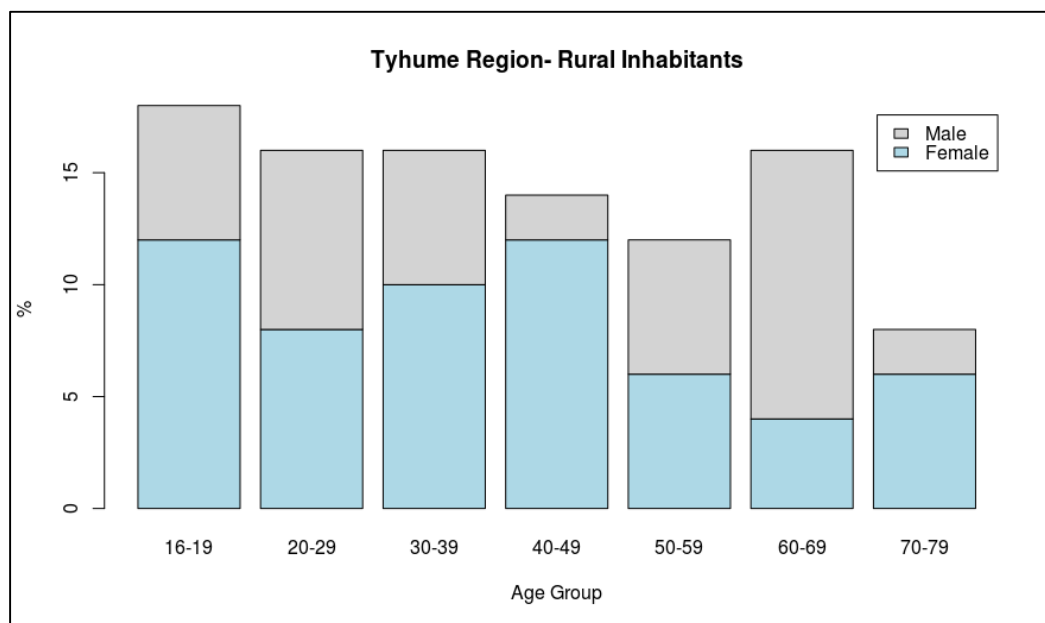


Figure 3-1: Rural Dwellers in Tyhume



The graph shows, based on gender, the type of people that leave in the Tyhume area. Although in the 60s males appear to be dominant, the people who look after the families are females. It is worth mentioning that, based on the observations during the study, these families are mostly dominated by females who, in most cases, have responsibilities of looking after young babies. It was also discovered that teenagers dominate the whole population as each family has more than one teenagers who in most cases are females. Another important point to understand about this area is the level of education as depicted on Figure 3-2 below.

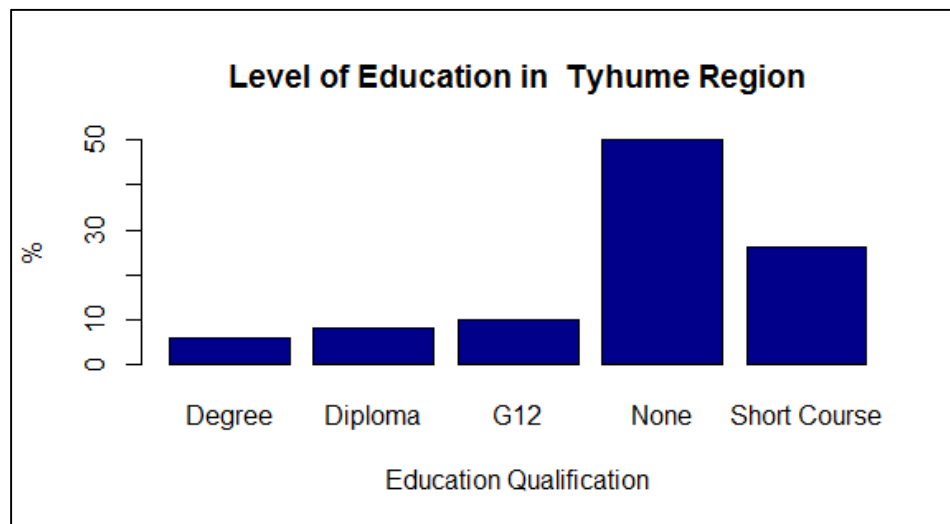


Figure 3-2 : Level of education in Tyhume

About 50% of people from this sample population do not have any certificate that qualifies them for a specific type a job. There are a number of contributing factors to this. As can be seen from Figure 3-1 that teenagers dominate this region, the same teenagers are still pursuing their high school educations. A few adult women who participated indicated that access to education in their time was not a simple process as it is now, especially in the areas where they grew up. About 18% of the sample population have at least short course certificates that enable them to be considered for low paying jobs. As education qualification becomes advanced, the number of people who have a qualification decreases as can be seen that only about 5% of participants have a degree qualification. If there is such a low level of education this area, how do these people earn their living? The graph on Figure 3-3 gives an overview of the employment rate in this area.

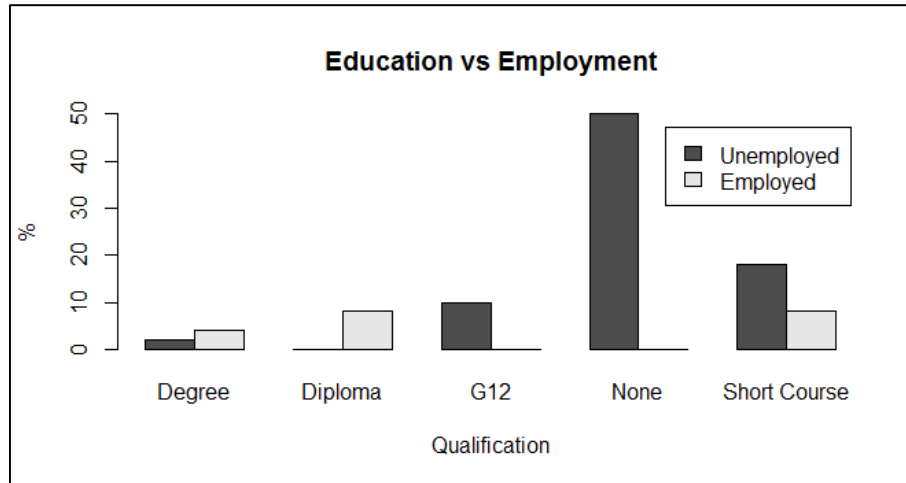


Figure 3-3: Education vs. Employment

It is evident that people who do not possess any kind of qualification are already swallowed by unemployment as depicted. It is again noticeable that, Grade 12/ Matric is still not enough to afford these people employment. Out of 18% of people who have at least short course certificates, about 50% of them still do not have jobs. Although the number of people who either have a diploma or a degree is very low, it's only in those qualifications where employment seems to be a little up. Figure 3-4 summarizes the overall employment in the Thyme region.

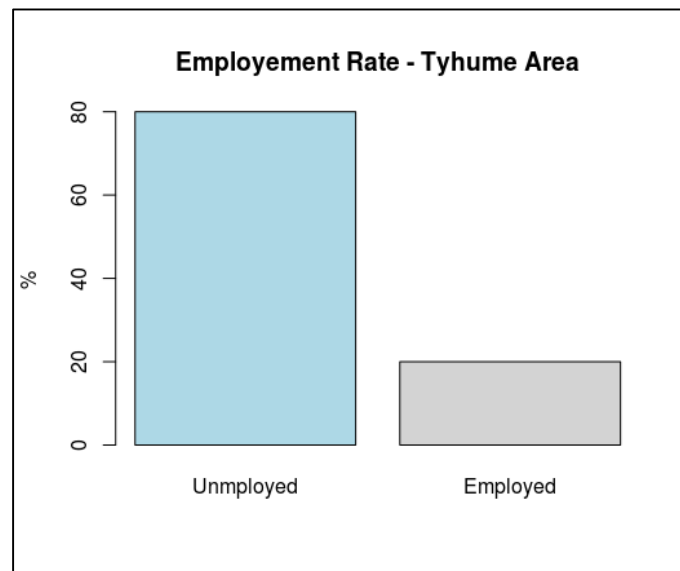


Figure 3-4: Employment Rate

The graph clearly portrays the rate of employment in the chosen area. Only about 20% of people are employed while 80% is unemployed. It is also essential to note that, the 80% also comprises

people who have more than just a matric certificate but also short courses. At this point, one can conclude on this aspect to say, based on the two previous graphs, the employment rate is very low in this rural area. Like any other rural area, dwellers in this area continue to visit urban based service providers on a day-to-day basis for various reasons. It is important for this research to unveil these service providers in order to enlighten the concerned stakeholders as they may wish to improve development and service provisioning. Figure 3-5 depicts the rankings of urbanized service providers (accepts for supermarkets) that are visited by rural dwellers as obtained from the study.

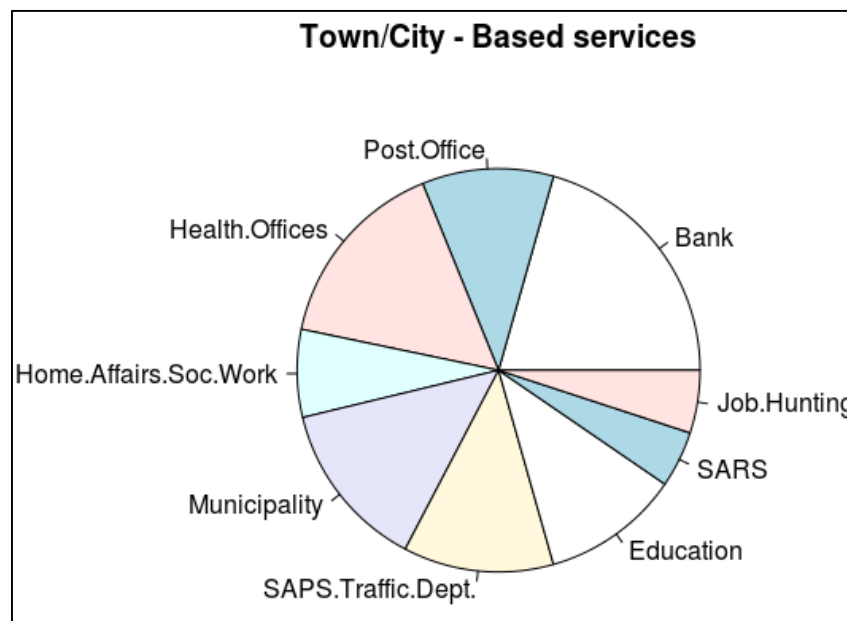


Figure 3-5: Services

Banks are shining brightly at top of the rank as the most visited service provider with a score of about 82% followed by health centers with a score of 66%, pharmacies included. One essential point to note here is that other than the banks, all other service providers are government offices. Municipal offices, police station, education institutions, and post offices all in that order appear to have a realistic number of rural visitors on day to day basis as they score more than 42% individually. Although job hunting is not a service or a service provider per se, it is, however, something that requires individuals to move from their homes to town or cities and interact with various service providers to negotiate employment.

### **3.4 Data Analysis Summary**

This section has provided a clear vision of the lifestyle in this rural community. The community is mostly dominated by teenagers who are mostly female in families that are looked after in most cases by a woman. Both the level of education and employment are very low in this community. The dwellers of this community continue to frequently consult their urbanized service providers for services that are very crucial for their living. Although this information is based on sample data, it is then believed that, if there exist living conditions such as these witnessed in this community, there are other communities with same conditions or even worse. It is important that the concerned stakeholders are instantly informed of these living conditions. In order to best align this research to the needs of rural citizens and towards producing the intended results, it is ideal to derive all the use cases and scenarios of the desired prototypes from the information obtained from the study above.

### **3.5 Requirement Elicitation and System Design**

It must be noted that the aim of this research is to demonstrate technical implementation of system customization and interoperability as defined here within in the context of rural development and electronic government. In order to keep aligned with the context, a number system components and functionalities were derived from the results of the study conducted. These components were used in the development of prototypes that act as platforms for service provisioning.

#### **3.5.1 Research Scenarios**

##### **3.5.1.1 Scenario 1 – User Account Management System (UAMS)**

This scenario assumes that government offices have agreed on the establishment of a central system to store and manage citizen profile and documents. Citizens should be able to securely update their profile including documents.

User Requirements

- Create account
- Update profile

Functional requirement

- Create Account
- Update Account
- Manager Accounts (assign roles, activate and deactivate accounts)
- Search User Account

### **3.5.1.2 Scenario 2 – Custom Application**

This scenario assumes that government has made a decision to have one customizable system that can be used by all departments to publish or disseminate government public records or information.

#### **User Requirements**

- Browse public records online
- Filter records by department or organization

#### **Functional requirement**

- Add administrators
- Create organizational domain
- Create data records
- Publish records (for public use)
- Update records

### **3.5.1.3 Scenario 3 – Birth Certificate Application Manager**

This scenario assumes that the Department of Home Affairs (DHA) needs to timeously receive records of newborn babies straight from the Department of Health (DoH) immediately after birth. These records must be in a digital format that can easily be processed by DHA to create automatic applications for birth certificates without having mothers to physically go to the offices unless there is a need. The DHA needs to have access to citizen profile and documentation on the central systems. With this information, the department will create birth certificate application that will be processed and mothers will be able to view processing results online. The processing results will be whether or not the automatic application was successful.

#### **➤ User Requirements**

- Get submitted applications
- Get application status

- Functional requirements
  - Get list of newborn babies (automatic application)
  - Get maternal information
  - Process application
  - Get processed applications
  - Find application by parent (maternal) Id

#### **3.5.1.4 Scenario 4 – Maternity Management System**

This scenario assumes that the DoH wishes to create a secure entry point that can be used to transfer only records of newborn babies to home affairs. The department also wishes to start monitoring and getting information about maternity patients “mothers/mothers to be” during pregnancy. Patient must keep their profile/details updated on the central system. The DoH will then use information from the central system for patient registration. A patient who does not have a profile on the central systems should be assisted to create it as soon as they discover their pregnancy at the health centers. The patient should also be able to view the diagnosis information online anytime.

- User Requirements
  - Get pregnancy diagnosis records
- Functional requirements
  - Get list of newborn babies (automatic application)
  - Get maternal information
  - Process application

Scenario two (2) above is more concerned with the dissemination of information which can be aligned with the cataloging stage of electronic government. Scenario three (3) and four (4) are concerned with the integration of multiple systems to offer a transparent and reliable approach to performing each department’s office duties. Each scenario presented above has also been given a number of user and functional requirements that can be drawn from it. This ultimately gives an idea of systems or components that can be used in this research to implement and demonstrates systems customization, integration, and interoperability as key concepts in this work. With the user and functional requirements defined for each scenario, it is then safe to begin the design of the prototype as the requirement of this work. Below are some of the non-functional requirements that apply to almost all systems and components.

### **3.5.2 Non-Functional Requirement**

#### **➤ Security**

According to the literature presented on this work, security is a critical aspect of ICT tools or services. As indicated earlier, security in this context is the ability to protect information systems and contained information from unauthorized access which may result in damages, theft, disclosure, misuse and any other undesirable action as suggested by Andress (2014). Since security is perceived in this research as being the state or quality of the system and not any form of a transaction or a service which would make it functionality, it is then categorized as a non-functional requirement.

#### **➤ Interoperability**

Interoperability in this research is defined as the ability of two or more systems to seamlessly share and use information in an automated manner. As the overall scope of this research suggests, systems design for use in the context of this work should be able to share information with other systems.

#### **➤ Usability**

It is important to ensure that the system can flexibly be used by the intended users to complete specific tasks without confusion. This again is not a functionality of the system but rather the extent to which users can easily use the system to perform tasks hence it is put here as a non-functional requirement.

#### **➤ Availability**

The system presented in this work offer a number of operations or functionalities for users while some also offer functionalities and operation for use by a number of other systems. It is important to make sure that each system or component is up, running and available when it is needed either by users or other systems.

## **3.6 Conclusion**

This chapter has given a detailed analysis of the data obtained from the study that aimed at revealing the state of rural communities as part of a sub-objective of this research. Scenarios were

then drawn from that that analysis to provide an example that can be followed to proceed with design and development of the intended prototype. The idea was to draw these scenarios based on the actual services that would make differences in existing problems that rural communities are faced with currently. Each scenario was then coupled with a number of user and functional requirements that would need to be included in the design. A point to note here is that scenarios are drawn from existing service providers that rural dwellers need to interact with mostly. This would help produce a prototype that is also aligned with the rural development and electronic government context. Non-functional requirements are also defined as they apply equally on all the scenarios.



## **Chapter Four: System Design and Technologies**

### **4.1 System Modeling**

Before giving details about system modeling, it is important to present the systems and components as derived from the scenarios defined in chapter three. The choice of each component/system or use case used in this research was motivated by the results of the study that was conducted from the specified rural communities as discussed in the previous chapter. Part of the main objective of this research is to produce a customizable system as a means to cater for the cataloging stage of e-government implementation as a result, a prototype system referred to as “custom-app” or “custom-system” is developed to demonstrate how various departments can publish public information with just one system catering for various domains. It is believed in this research that a system of this nature would reduce costs of service delivery in rural marginalized communities. One point to note here is that there are websites that are explicitly for information sharing and most of them are not yet at the level of providing electronic transactions. It is not ideal to define user account component for each of these systems or components. It could be best to have one central user account management system that is accessible from all the systems that are meant to provide services to the public hence user account management system (UAMS) is presented. This would avoid inconsistencies in citizen information on the systems hosted in respective offices.

From time to time people have to consult municipal offices, SAPS offices and also traffic Department for documents that are required by other offices in order for them to get help. For that reason, a document management system (DMS) is used as one of the components to implement and demonstrate interoperability as per the objective of the research. The last two systems used to demonstrate system interoperability assume the cases of Department of Health (DoH) and Department of Home Affairs (DHA). These two systems both make use of UAMS to substantiate system integration as a key concept of this research.

#### **4.1.1 Custom System-Use cases**

Use case diagrams are normally used to present a list of actions or event steps that define interactions or relationships among various objects (ISO/IEC and IEEE, 2010). Figure 4-2 shows such kind of interaction for the design of the custom-system presented in this research. The use

case constitutes four main operations and two types of users, namely, admin user and client user. These operations are actually services that the custom system offer. Users are able to initiate a session by logging in the system to perform whatever action they need depending on their roles. Users with administration privileges are able to create new domains under which entities can be created. A domain can represent one department and entities in each domain can represent a record of information within that department. This means that each entity must be defined inside a certain domain. The “Manage Entities” use case groups a number of basic use cases related to each entity including the creation and updates. Administrators also have an obligation to manage contents for entities which they monitor. Managing content includes adding new content and view existing content. From the “Manage Content” use case which is a group of other basic use cases shows that the client user can only have access to viewing existing content. Since this is a custom system which is also the central access point, it is also possible to search for someone from the system although the actual search will be performed by the adjacent system which is User Account Management System (UAMS).

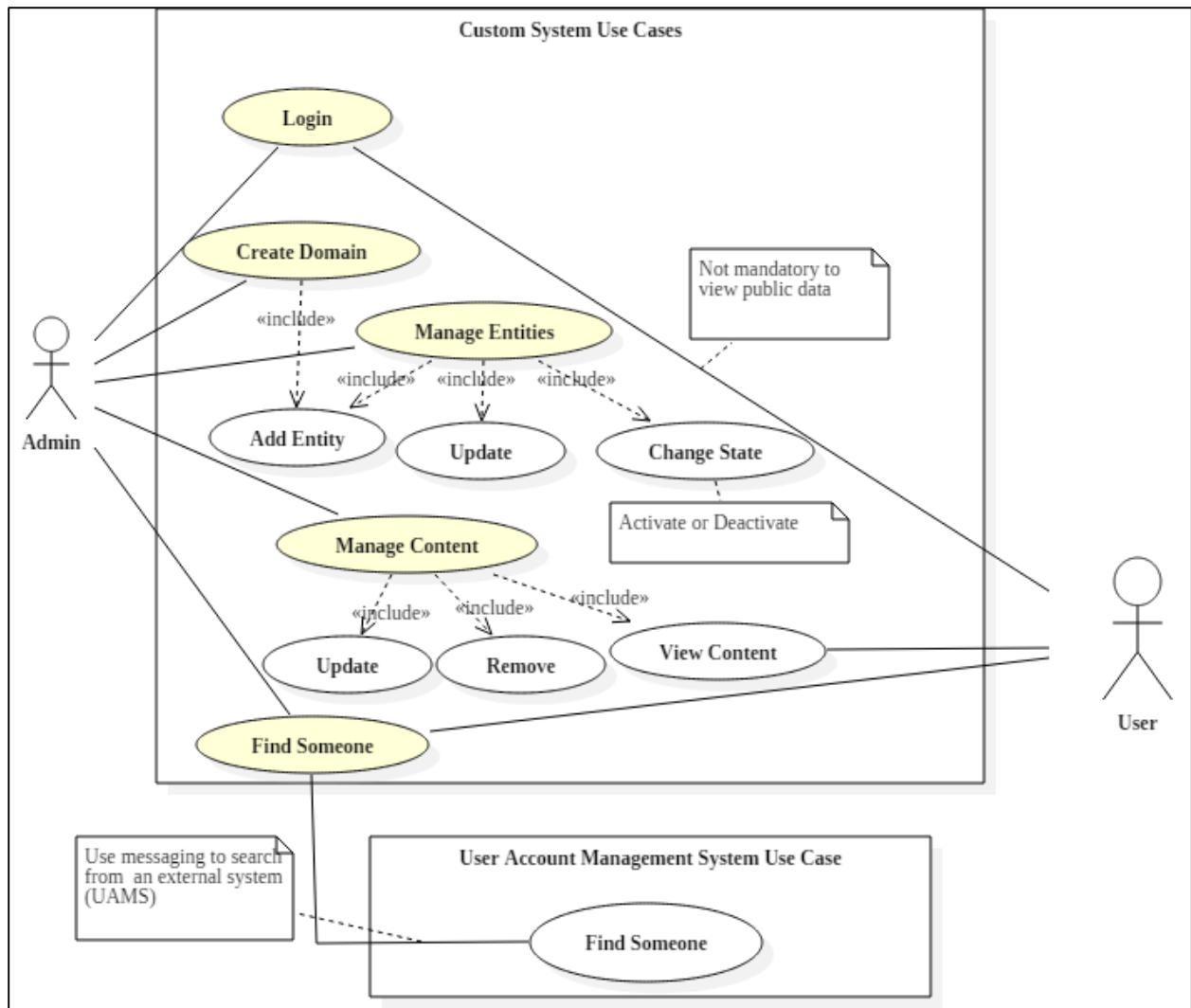


Figure 4-1: Custom System Use Case

#### 4.1.2 User Account Management System (UAMS) - Use Case

Figure 4-2 below shows some of the operations that can be performed on UAMS. The targeted users (rural citizens) can be able to apply for system user accounts that they can be used in the event that some services available require authentication. In cases where users need to make some changes on their account, they can do so using the “manage account” use case which includes user information updates and changing account status. Administrators can also assign roles to client users. Lastly, the “Find Someone” use case makes it possible to search for anyone and get basic information about that particular person.

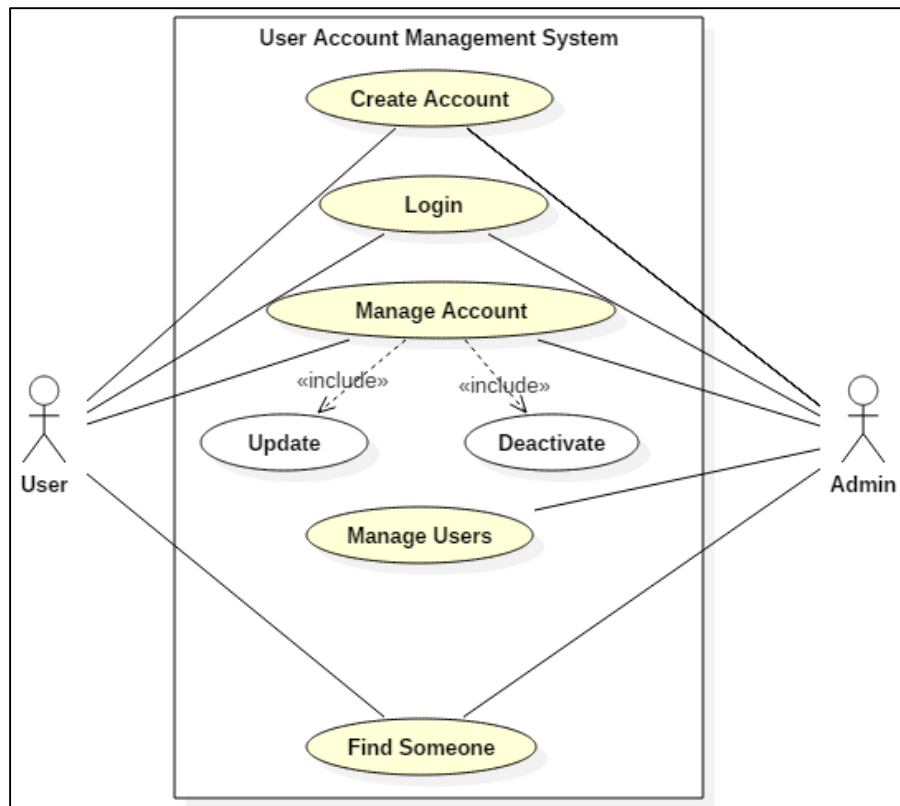


Figure 4-2: Account Management System Use Case

### 4.1.3 Domain Creation - Sequence Diagram

The use cases presented above are the main activities that the system can provide. Further on we can explore a sequence of action events as each use case is executed. To begin with, Figure 4-3: *Create Domain Sequence Diagram* shows a sequence diagram for creating a new domain on the custom system. This begins when an administrator sends a “create” request to the system using the system’s user interface presented on the administrator’s web browser.

It is a good idea to have all forms checked and validated on the front side before they are submitted for processing. For that reason, as shown in the sequence diagram, the form should be checked before it gets sent to the controller which is responsible for processing request by calling the service that is injected into it. When the processing occurs and the logic is executed, JPA repository interface should be invoked in order to be able to persist data. The controller will always be aware of the output of the processing phases and it will take the output together with the view name which it should be presented on and it will dispatch it back to dispatcher servlet where the view

resolver will be used to find the appropriate view. The output data will then be rendered on the resolved view for display to the user.

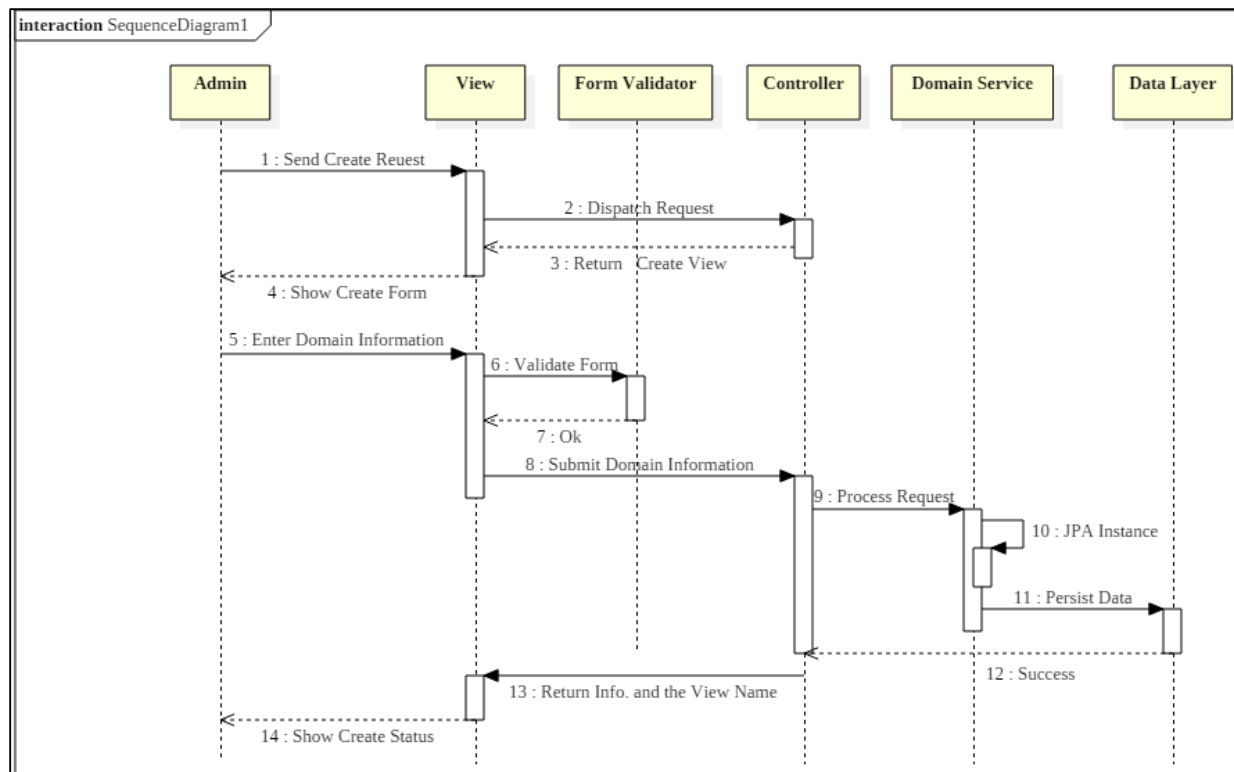


Figure 4-3: Create Domain Sequence Diagram

#### 4.1.4 Entity Creation - Sequence Diagram

Creating an entity can be part of two sets of action events as indicated on the custom system use cases. A primary entity can be mandatorily created in the process of creating a new domain. This is a rule that ensures that there are no domains that reside without entities, this means that for a domain to be successfully created, at least entity must be defined. More entities can always be added on existing domains to grow them. This can be archived on the “Manage Domain” use case by adding a new entity.

Figure 4-4 presents a list of action events and interactions that occur when creating an entity. In order for an entity to be created, the administrator must first select the domain in which the entity will reside. When inside the domain, a request to create an entity must be initiated using the system’s view presented. That request will then be forwarded to the controller which returns a

view that accepts entity information for entity creation. Again here the details between the controller and view are omitted as discussed in the view's layer of the detailed architecture. When entity details are submitted they are checked first on the view side for validation so that the controller does not receive invalid input. Upon validation success the details are then submitted to a controller where the entity service injected on that controller will be used for the actual processing of the entity creation request.

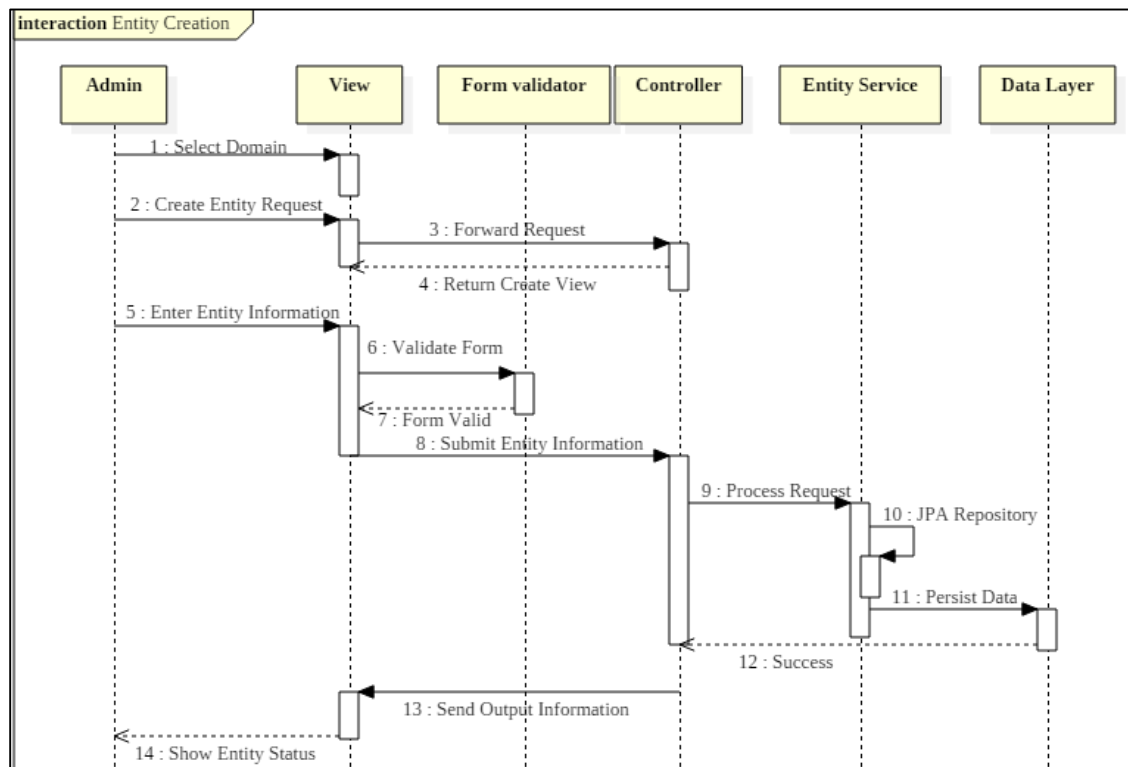


Figure 4-4: Create Entity Sequence Diagram

The entity service will in the process invoke a spring JPA repository that is used for persisting the services output which will be the new entity in this case. The results of whether an entity was successfully created or not will then be forwarded back until they reach the user's view.

#### 4.1.5 Maternity System

This system assumes a case of women during their maternity process until they give birth and later need to apply for a birth certificate. Below is the use case diagram for the system.

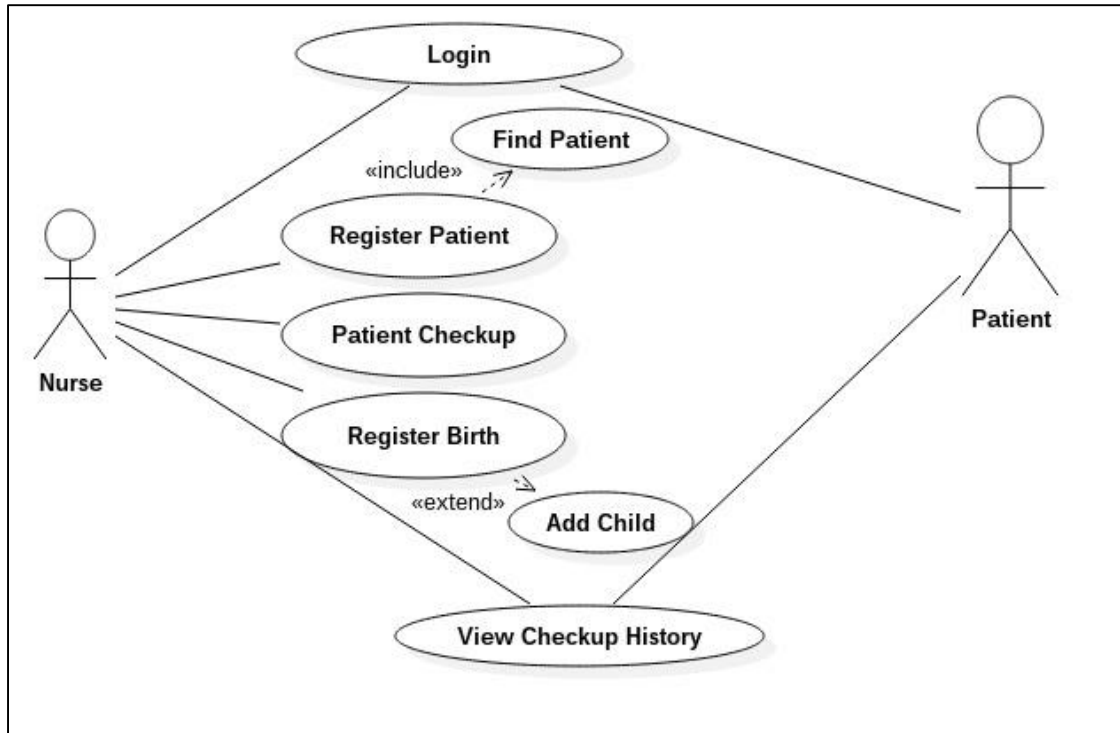


Figure 4-5: Maternity System

All services offered by system require users to log in. Nurses or any medical practitioner can be able to register patient as soon as they discover that they are pregnant. Registration is quite interesting as it requires the use of external systems to verify and validate the patient information. They would feed the system with patient's Id number or full name which the system will use to search for full details on user account management system through web service invocation. As patient visit health centers on monthly basis, the same system can be used to record the details of checkups to form a solid record of the patient's maternity process. The patient can always visit the system to view the records of their checkup. On the day of labor, when the patient gives birth, the system can be used again to capture all the details of the labor process including child's name and relevant details should the labor process be successful.

Although not important for now, there is also a messaging agent that is responsible for polling all birth/ labor records and sends them to another system responsible for birth certificate application. The sequence diagram in Figure 4-6 gives more details on how each patient is registered on the system.

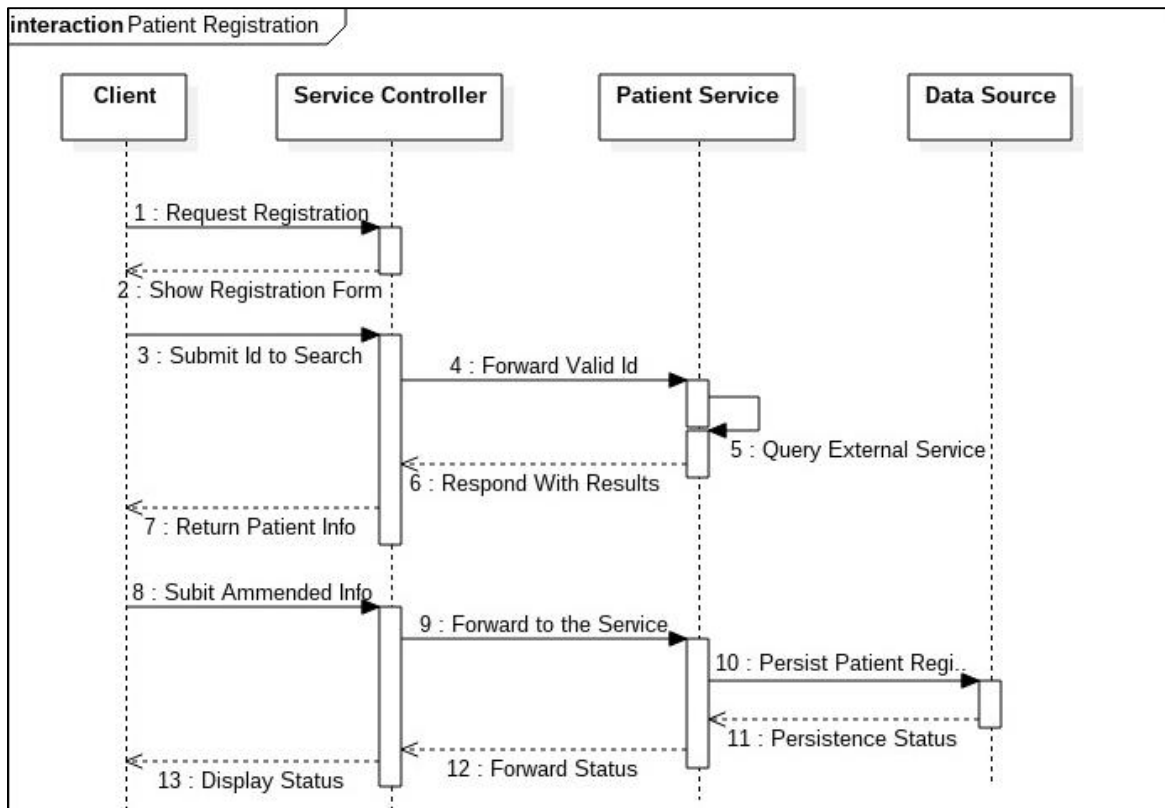


Figure 4-6: Patient Registration

To register patients, the practitioner sends patient's Id number to the system using a web form, passing all the validations. The Id reaches the web controller which is responsible for calling the actual service. The controller then calls a specific method in that service to request information about the specified Id. The response is then sent back by the method to the controller which will then, based on the results, decide what to send back to the user. If the controller returns the patient's details, the practitioner will then start adding attributes that are specific to the maternity registration and send it back to the controller. At this stage, the controller will disregard some of the user details and keep only the id, full name, and details specific to the registration. The resulting patient details are then sent to the data access service for persistence.

#### 4.1.6 Birth Certification Application

Birth certificate applications is a standalone application that assumes a specific role at the home affairs office. When the person responsible for managing birth certificate applications logs into this system, they can be able to get a list of all new unprocessed applications with details of



newborns and their parents and can immediately sort applications and pick them one by one for processing.

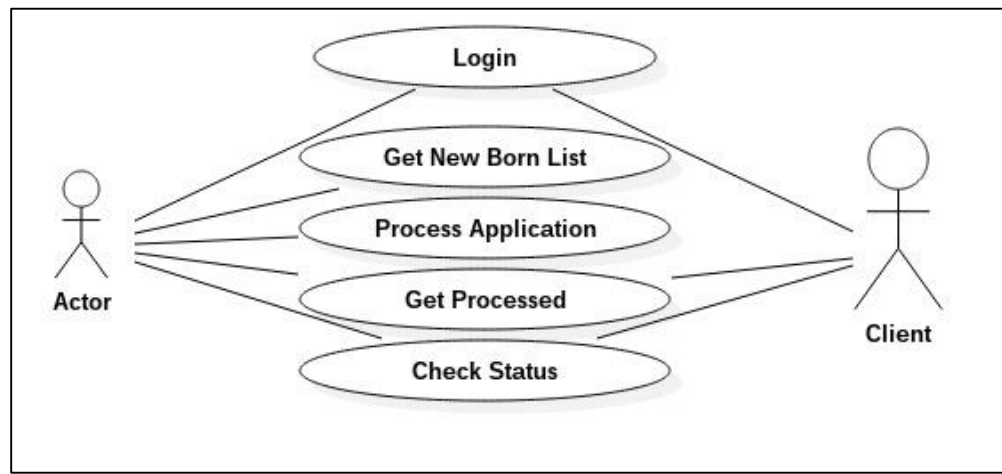


Figure 4-7: Birth Certificate System Use cases

The client on the other side which can be another system or a specific user can be able to get all processed applications that are ready for collection and also check status for individual applications. One thing that is not clear at this moment is where these applications come from. They are loaded into this system by a messaging agent that works polls the maternity system for DoH discussed earlier.

This section presents an overall design of the applications produced by this research. It is more than just a design because it specifically gives details of both architectural and detailed design and system modeling. System architecture and detailed design are understood and used in this research as defined in (ISO/IEC and IEEE, 2010). The modeling system used in this research based on, but not limited to interaction modeling.

## 4.2 System Design

### 4.2.1 Architectural Design

Figure 4-8 presents an architectural design followed in this research. The first thing that the architecture requires is an in-depth understanding of the frameworks to be used to develop applications that will later be integrated. This layer also includes the preferred or selected programming languages, platforms and environments. Interoperability is one crucial part of this

architecture as it requires agreed-upon data exchanges formats such as XML and JSON, naming conversion and exchange of some common artifacts.

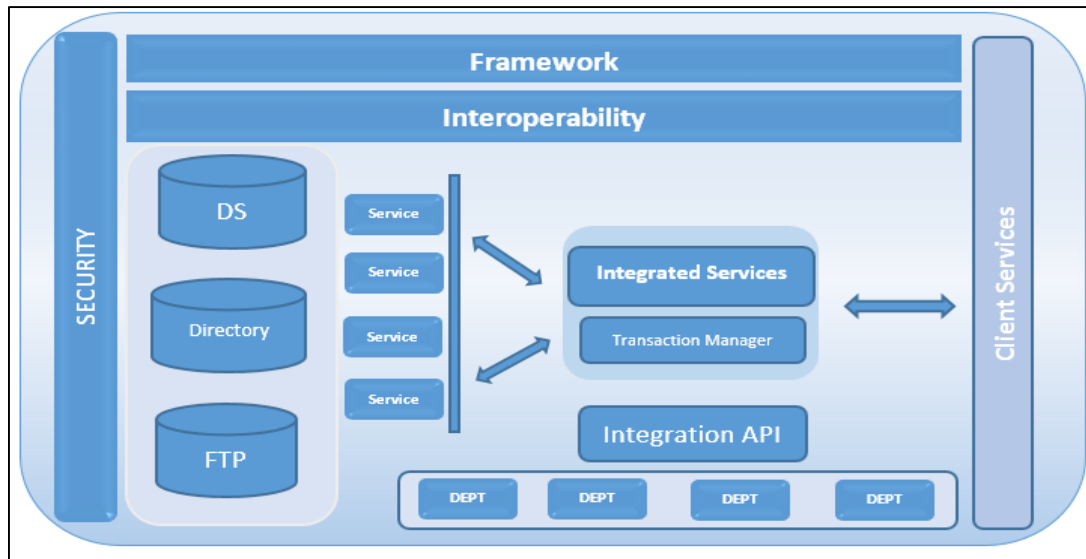


Figure 4-8: Architectural Design

Although not the main focus of this research, security is also a very crucial aspect and hence the architecture suggests that security is also defined in the design and development of systems of this nature. The data source layer defines components such as databases, the directory database, and FTP servers. Services represent certain functionalities offered by any organizations of the departments. The transaction manager is a part of a layer where all these services are unified and accessed by their intended consumers or clients. It could be a self-contained application that is responsible for querying underlying applications and present the results in a unified manner or an ESB environment that hosts and avails all services for consumption by the intended users or other systems. All the layers that the architecture presents apply to all departments or any relevant service provider. One other important thing is the identification of all services that each department provides and such services must be presented in this architecture. There should be no limit in the number of services that each department or organization can have. The architecture again does not force system modules or services to reside exactly on the same site or base with the main integration application (e.g. transaction manager), this then allows the architecture to be able to connect to disparate remote systems. It is now evident that this architecture is mostly concerned about the services that each department or organization offers to the public.

A more detailed architecture is presented in Figure 4-9 with more information regarding how services or system modules should interact and how the messaging between them is managed.

#### 4.2.2 Detailed Design

Figure 4-9 below presents a more detailed design the specification of communication and messaging among system components. Systems and components or modules need to implement any of the communication or invocation method specified, namely, rest service, soap services or messaging mechanisms such as JMS. In the case of web services, Apache CXF is found to be a relevant web service framework hence it is used in this research.

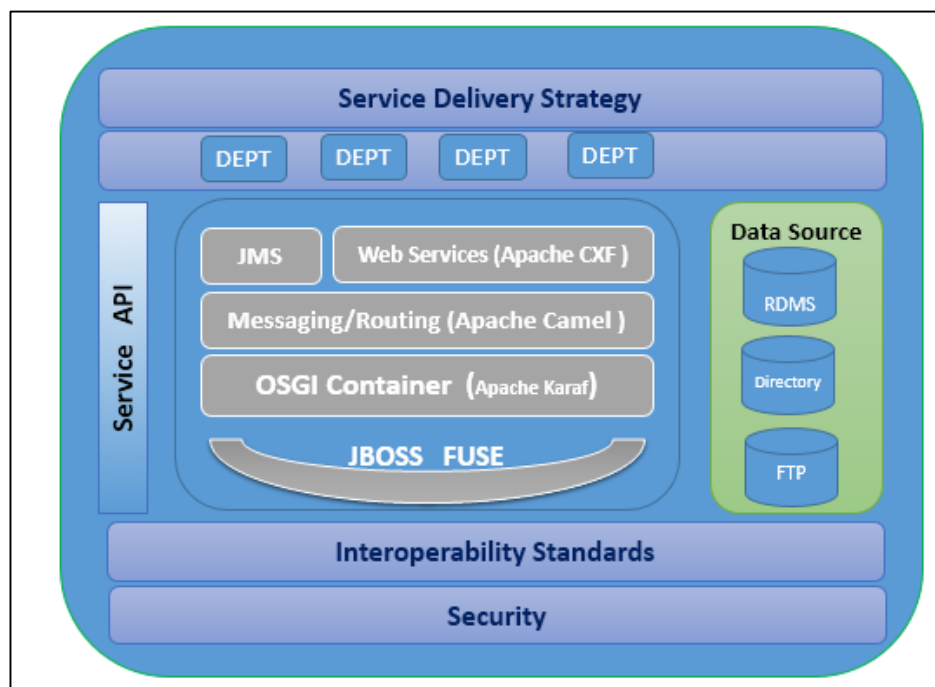


Figure 4-9: Detailed Design

Apache camel is then used to define message or data routing mechanisms for both external and internal systems as represented by endpoints. The endpoint in this case is an object that has a definition of properties such as URIs and other things that describe a web resource or service. It is these endpoints that are used to define routes to transport messages across various systems. The services, endpoint and routes can then be deployed in an OSGI container such as Apache KARAF where they can be accessible to other services or clients.

### 4.2.3 Spring Application

In many cases, as scenarios presented earlier represent a certain system, although these systems provide means to communicate with other systems, they are also meant to provide full functionality within the respective domain. This means that each system must be usable internally to support department's duties before it can communicate with others. Java EE and spring framework are two chosen frameworks to implement the needed web applications in this research.

Figure 4-10 below shows the interaction of different parts of a spring application in the context of this research. The diagram can be broken down into four layers although not all systems or components need to implement those layers. Firstly, there is a view layer (presentation) which is made of dispatcher servlet, controller, view resolver and the actual view.

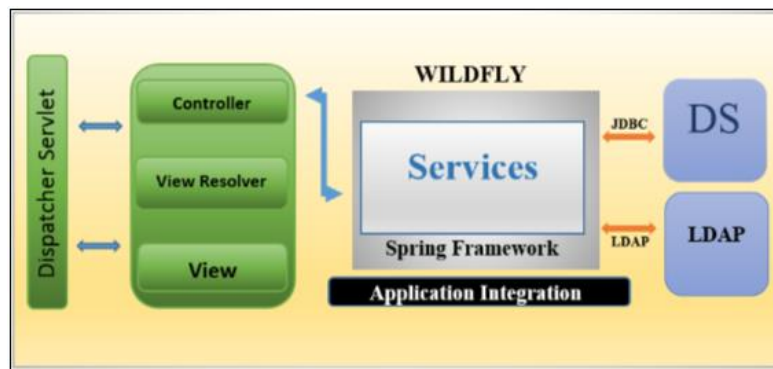


Figure 4-10 Spring Application

Secondly, there is service (Logic) layer followed by data layer which constitutes both relational and directory data sources. Lastly, there is an integration layer which is not of interest for now. Although it is not the main focus of the research, one more layer that is very crucial for almost all systems and components is a security layer, which cuts across from the presentation layer up to the data source layer.

#### 4.2.3.1 Presentation layer

This layer is responsible for presenting relevant data to the web clients. It is based on spring Model-View-Controller (MVC) design. The key actor in this phase is the spring dispatcher servlet which is responsible for accepting requests and return appropriate responses. As it receives the request it then consults the handler mapping to find out which controller can process that request and it will eventually dispatch that request to the appropriate controller. It is the controller that has knowledge

(through dependency injection) of which service should be invoked for the actual processing of the request. The controller then returns a logical view name and output data (from processing) to the dispatcher servlet which relies on the view resolver to find the actual view. The data is then rendered in the view and then sent back to the dispatcher servlet which it returns to the client as a response.

#### **4.2.3.2 Service Layer**

This layer comprises a number of important interfaces, classes and various XML configurations that are used to implement the service logic for different use cases as will later be seen in this chapter. While some the interfaces are used as Data access objects (DAO) for the actual service implementation, some are used to expose spring data JPA in the service implementation classes as means of persisting data. There are also beans that service implementation classes require in order for them to be complete and such beans are defined in application context written in XML. These services are then called or injected on appropriate controllers to perform their intended logic upon request.

#### **4.2.3.3 Data Layer**

The data layer is concerned about the storage and retrieval of user information and the actual data that is communicated among various components. LDAP is used to store directory information such as user profiles and role information. This information is then kept on an LDAP server that runs separately from the application. Relation database server (MySQL) on the other hand is used to store any other data used by the application.

#### **4.2.3.4 Integration Layer**

This is the entry point for any other system that wishes to use information defined in this systems. The way this layer is designed is such that it only exposes the information that is meant to be shared across the ICT architecture.

### **4.2.4 Java Enterprise Edition (Java EE) - Enterprise Java Beans (EJBs)**

The second choice of application development framework or specification in this research is Java EE. The implementation of services in this approach uses EJBs and is different from that of spring

frameworks. EJBs in their nature are container managed beans where business or service logic is encapsulated in a flexible, secure and portable manner. Document management system implemented for the purpose of this research is designed and developed following this approach.

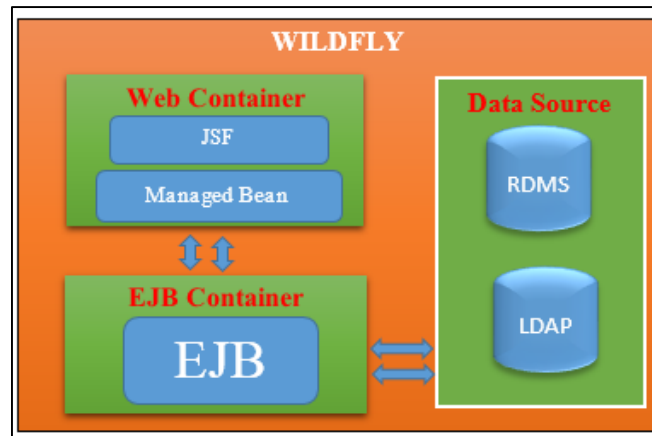


Figure 4-11: EJB Application

As can be seen in Figure 4-11, the way EJBs work is such that the actual logic is hosted inside an EJB container that in this case is provided by WILDFLY. The application server also offers a web container that hosts the web content and is responsible for rendering pages as requested by the user. Managed beans are used to manage request-response to and from the EJB container through dependency injection. It is the EJB and its context that knows the details of data source connection in cases where service logic needs to persist or retrieve data.

### 4.3 Tools and Technologies

In order to fully understand the various tools and technologies used in this research, it is best to first understand the development environment used. The development environment was successfully set up in a lab machine running Linux 32 bit operating system. The programming language of choice in this research is java, chosen on the basis of its understandability by the researchers as that would reduce the learning curve on programming languages and enable more focus in the context of the research. Other related technologies were selected after a lengthy practice and comparison of different approaches to developing applications. This practice and comparison were based on programming models or frameworks and web application servers. Spring Framework, JBOSS Seam, and Enterprise Java Beans (JEE/EJBs) were used to create basic applications with the aim of noting, among other aspects the advantages, public support and

simplicity. Each application could then be deployed on three application servers which included WILFLY, Tomcat, and glassfish. Different IDEs were tried for different programming models or frameworks during the practice.

Spring Framework was adopted as the main framework for this research in Spring Tool Suite (STS) and JBoss Developer Studio development platforms. WILDLFY on the other hand was adopted as the application server to host all applications. This has given an overview of some of the main technologies use in this research.

Each of the systems was implemented with the idea of exposing an entry point for other systems to consume data through web services or a JMS support. Apache CXF was selected as a framework of choice for implementing web services. Apache Camel was also selected as a system integration framework of choice. Below is the list of the tools used in this research.

- Linux Ubuntu 12.04 LTS 32 Bit OS.
- Spring Tool Suite 3.7 IDE
- Spring Framework 3.2.3
- JBoss Developer Studio
- WILDFLY 8.2.1 Final
- Java Development Kit (JDK) 1.7
- EJB
- JMS
- MySQL
- LDAP
- JSP
- JavaScript
- JSON
- Apache Camel
- Apache CXF

## **4.4 Conclusion**

This chapter began with the introduction of systems and components to model and design as derived from the defined scenarios. An interactional modeling approach was then used to model and desired systems. An architectural design was presented to give an understanding of the overall interaction between participating components of the desired platform. The detailed design introduces tool and technology specification in the overall design. Spring framework and EJBs, came out as two choices to develop the intended systems although a room for other styles or framework was kept.



## **Chapter Five: System Implementation**

### **5.1 Introduction**

The literature on the e-government studies has shown that implementation of e-government requires a solid understanding of e-government phases (Seifert, 2003; Dayanidhi, 2005). The first phase known as cataloging is mostly concerned about dissemination of information. This includes making sure that all necessary information that citizens need from government offices is available together with documents such as forms that can be downloaded (Seifert, 2003; Dayanidhi, 2005). Other stages are concerned about electronic transactions and different types of system integration. Having understood the theoretical explanation of all these e-government stages which are known to be contributing factors towards a successful implementation of electronic government and public service delivery, it was then necessary to investigate and demonstrate these stages at a technical level. This chapter, therefore, presents the implementation of prototypes that demonstrate these stages together with other concepts of interest such as interoperability and customizability at a technical level. As mentioned earlier a number of scenarios are defined in order to properly demonstrate all the key concepts this research is concerned about. Some of the scenarios or applications used in this research follow and use similar technologies.

### **5.2 Dissemination of information**

The design and implementation of the prototype to demonstrate this stage of e-government also encompass the concept of customizability. This is so as to avoid creating separate applications to keep similar information that could be easily managed under one system with a lot more advantages at minimal costs. One of the advantages is having a single point of access for all public information which could also be reusable and shared among other systems that need to use the same information or records. As there would be a need for a number of departments disseminating information, the prototype used to for this case uses the concept of “domain” to represent a department and all the departmental records would be inside that domain.

#### **5.2.1 Domain Creation**

To create a domain, an application must be made with all the necessary information about the

required domain as shown in the listing below.

```
@Entity
@XmlAccessorType(XmlAccessType.FIELD)
public class Domain implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int domainId;
    private String domainName;
    private String orgName;
    private String contactDetails;
    private String creator;
    private String category;|
}
```

Listing 5-1: Domain Model

These properties are then used to identify each domain for a number of cases such as the organization or department. The request to create a particular domain will then be made from the client which is implemented spring MVC. Listing 5-2 shows a spring controller that accepts the properties above and call the specific service create an actual domain.

```
@RequestMapping(value = "createdomain", method = RequestMethod.GET)
public ModelAndView createDomain(HttpServletRequest request,
    HttpServletResponse response) {
    Domain domain = new Domain();
    ModelAndView model = new ModelAndView("home");
    domain.setCreator(request.getParameter("domainCreator"));
    domain.setDomainName(request.getParameter("domainName"));
    domain.setOrgName(request.getParameter("domainOrg"));
    domain.setContactDetails(request.getParameter("contact"));
    domain.setCategory(request.getParameter("category"));
    domain.setDescription(request.getParameter("description"));
    if (isDomainModelValid(domain)) {
        SystemMessage sysResponse = domainService.createDomain(domain);
        if (!sysResponse.hasException() || sysResponse.isSuccess()) {
            model.addObject("message", "success");
            model.addObject("domain", domain);
        } else {
            model.addObject("message", "Operation failed \n:" + sysResponse.getMessages());
        }
    } else {
        model.addObject("message", "Invalid domain details supplied");
    }
    return model;
}
```

Listing 5-2: Domain Controller

The controller receives an HTTP request with all properties which it checks if they are valid and calls a domain service to persist the details. The domain service on this spring controller is injected using spring “Autowired” annotation. The domain service method to create domain returns a system message object which is a customized object to collect system messages and/or exception

throughout the service calls. As can be seen in Listing 5-1, the domain model is a JAVA Persistence API (JPA) entity or object. Shown below is the implementation of the actual service that saves r persists this object.

```
public class DBCreator {
    @Autowired
    private DomainRepo domainRepo;
    ApplicationContext cxt = null;
    final String contextPath = "classpath:META-INF/applicationContext.xml";

    public SystemMessage create(Domain domainObject) {
        SystemMessage message = new SystemMessage();
        try {
            cxt = new ClassPathXmlApplicationContext(contextPath);
            DomainJDBC jdbcTemplate = (DomainJDBC) cxt.getBean("DomainJDBC");
            JdbcTemplate jdbc = new JdbcTemplate(jdbcTemplate.getDataSource());
            Domain createdDomain = domainRepo.save(domainObject);
            if (createdDomain != null) {
                String statement = new DBCreateStatement(
                    domainObject.getDomainName()).Get();
                jdbc.execute(statement);
            }
        } catch (Exception e) {
            HashMap<String, String> map = new HashMap<String, String>();
            map.put(e.getClass().getName() + "", e.getMessage() + "");
            message.setMessages(map);
            message.HasException(true);
        }
        return message;
    }
}
```

Listing 5-3: Domain DB Creator Service

The domain service also calls the DBCreator service as shown on listing above via spring's Autowired annotation. As indicated earlier that the domain model is JPA entity, it is then saved using spring data JPA interface which the developer does not necessarily have to worry about its implementation and can use it as depicted in Listing 5-4.

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import webFrameApp.entites.Domain;

@Repository
public interface DomainRepo extends JpaRepository<Domain, Integer> {
}
```

Listing 5-4 Domain Service Repository - Spring Data JPA

In order for this interface to perform the necessary transactions, it only needs to be provided with the JPA entity and the data type of the identifier property. Returning back to the DBCreator service, when the JPA entity is successfully saved, the service begins creating the first database where all domains records (tables) will reside. The way it does this is by calling the JDBC bean that is defined in the spring XML configuration file for accessing connection to the underlying MySQL

server. This means that to have a domain successfully created, it must exist as an entity to provide more information about the domain, and it must have a database created for all its records.

### 5.2.2 Domain Entity – organizational records

Domain entities are used to keep the data inside a particular domain. Listing 5-5 below shows a model with all properties that are used to define an entity.

```
@Entity
@XmlAccessorType(XmlAccessType.FIELD)
public class OrgEntity implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer eid;
    @Column(unique = true)
    private String name;
    private String domain;
    private String[] attributes;
    private String[] types;
    private String[] other;
    private String[] optionValues;
```

Listing 5-5: Organizational Entity

Most properties of this model are defined as arrays of strings on the basis that different entities can have a different number of attributes. Each attribute will, in turn, have a corresponding datatype and other constraints that may be required for a specific entity. Another important point to mention here is that again this model represents a JPA entity that can be persisted the same way domain entities are persisted. Furthermore, the inputs or values of the properties of the models are supplied using an MVC controller where they get forwarded to a service that persists the JPA entity thereafter creating a table to keep data for that entity. Shown in Listing 5-6 is a service that deals with the part that inserts data on these entities.

```

public boolean saveContent(String sql, String[] data, String domainName,
    String entityName) {
    JdbcTemplate jdbc = new JdbcTemplate(jdbcTemplate.getDataSource());
    Connection con = null;
    String newTable = "tbl_" + entityName;
    String catalog = domainName + " db";
    System.out.println("the Catalog is: " + catalog);
    System.out.println("the table is: " + newTable);
    int results = 0;
    try {
        con = jdbc.getDataSource().getConnection();
        con.setCatalog(catalog);
        PreparedStatement statement = con.prepareStatement(sql);
        for (int x = 1; x <= data.length; x++) {
            statement.setObject(x, data[x - 1]);
        }
        results = statement.executeUpdate();

        con.close();
    } catch (SQLException e) {
        System.out.println("There is a problem with sql Connection");
        e.printStackTrace();
    }
    return results == 1 ? true : false;
}

```

Listing 5-6: Save Content

This method takes in an SQL query string, an array of values, domain and entity name and uses these to prepare a complete SQL statement that will be executed by JDBC. The method only gets called when all the attributes for the active entity are retrieved and used to prepare the SQL query string. All the method does is just set values for the prepared statement using a for-loop and then insert the data on the respective entity in a relative domain.

Retrieving data from an entity is quite an easy task than updating or adding new content. Listing 5-7 shows how to get content for a specific entity in a particular domain.

```

@RequestMapping(value = "web/viewcontent", method = RequestMethod.GET)
public ModelAndView viewContent(HttpServletRequest request) {
    ModelAndView model = new ModelAndView("viewcontent");
    String getDataFrom = request.getParameter("entity");
    List<OrgEntity> ents = OrgEntityService.findByName(getDataFrom);
    OrgEntity ent = ents.get(0);
    String domain = ent.getDomain();
    contentLoader loader = new contentLoader();
    String[] columns = loader.getCoolumn(domain, getDataFrom);
    List<List<Object>> allData = loader.getData(domain, getDataFrom);
    model.addObject("labels", columns);
    model.addObject("data", allData);
    return model;
}

```

Listing 5-7: View Content Controller

The controller only takes in the entity name as HTTP request parameter, it uses that entity name to get the entity properties and its domain. The controller calls another class that has a method to return the data as a list of rows with multiple columns as shown in Listing 5-8. The data is then sent to the spring view where it is manipulated and displayed.

```

public List<List<Object>> getData(String domain, String entity) {
    String catalog = domain.trim() + dbSuffix;
    String newTable = tablePrefix + entity;
    String sql = "Select * from " + newTable;
    contentLoader cl = new contentLoader();
    List<List<Object>> allData = new ArrayList<List<Object>>();
    try {
        Connection connect = jdbc.getDataSource().getConnection();
        connect.setCatalog(catalog);
        Statement statement = connect.createStatement();
        ResultSet resultSet = statement.executeQuery(sql);
        int x = cl.getColumnCount(domain, entity);
        int columns = resultSet.getMetaData().getColumnCount();
        while (resultSet.next()) {
            ArrayList<Object> list = new ArrayList<Object>(columns);
            for (int z = 1; z <= x; z++) {
                list.add(resultSet.getObject(z));
            }
            allData.add(list);
        }
        connect.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return allData;
}

```

Listing 5-8: Load Content - Service Method

The method gets a JDBC connection from a spring bean defined in the application context. It uses this connection to execute the SQL query statement in relevant connection catalog.

So far discussed in this chapter is how organizational records are created and grouped together in a specific domain. It should be noted that a more proper administration of these domains and records will be discussed in the security section.

## 5.3 System Integration

This section provides the implementation of prototype systems that are used to demonstrate integrated electronic transactions and interoperability as key concepts of this research. It should be noted again that these concepts are linked in the context of this work with mostly the second stage e-government and service delivery.

### 5.3.1 User Account Management System

The user account management system (UAMS) is used in this research to provide a central platform for keeping the credentials and other details of all the users of the systems developed as part of this work. It is also developed separately to also offer other features such security (authentication and authorization) to be discussed later in this chapter and is also used to demonstrate interoperability as a key concept of this research. Shown in Listing 5-9 are some of the properties that are required to create an account for a user.

```

@XmlRootElement
public class Person implements Serializable {
    private String first_name;
    private String last_name;
    private String idNumber;
    private String gender;
    private String cell;
    private String email;
    private String username;
    private String password;
    private String orgUnit;
    private String dn;
    private String member;
}

```

Listing 5-9: User Account Model

This model does not have JPA annotations like all other models seen above and this is because it is designed to only hold properties that will be persisted on LDAP server. A controller that makes use of this model to create an account is shown in Listing 5-10.

```

@RequestMapping(value = "home/submit_appl", method = RequestMethod.GET)
public String applic_submit(@ModelAttribute("Person") Person person,
    ModelMap model) {
    model.addAttribute("last_name", person.getLast_name());
    model.addAttribute("first_name", person.getFirst_name());
    model.addAttribute("gender", person.getGender());
    model.addAttribute("cell", person.getCell());
    model.addAttribute("email", person.getEmail());
    model.addAttribute("username", person.getUsername());
    model.addAttribute("password", person.getPassword());
    model.addAttribute("idNumber", person.getIdNumber());
    context = new ClassPathXmlApplicationContext(
        "classpath:META-INF/applicationContext.xml");
    BeanFactory factory = context;
    personInterface create = (personInterface) factory.getBean("ldapUser");
    create.createAccount(person);
    return "application_submission";
}

```

Listing 5-10: Create Account Controller

The first thing to look at on this controller is the use of model attribute annotation. This annotation allows easy mapping of the form fields on the view to the model. The second important part is the use of springs' application context to get the LDAP user bean that is cast to java interface that has operations that can be performed on the user account. In order to create an account, a connection to the underlying LDAP server must be established and be active. That connection is defined in the spring application context file. The implementation of "PersonInterface" class has a property that its value will be injected from this application context thereby allowing the class to have access to the underlying LDAP server. This means that "ldapuser" bean uses "PersonInterface" implementation and injects the LDAP connection on the LDAP template property defined in the implementation class. Listing 5-11 shows the code snippet for java class the implements the



interface mentioned above.

```
@Autowired
private LdapTemplate ldapTemplate;

public LdapTemplate getLdapTemp() {}

public void setLdapTemp(LdapTemplate ldapTemplate) {}

public void postConstruct() {}

@Override
@WebMethod
public void createAccount(@WebParam(name = "newperson") Person newPerson) {
    try {
        Attributes userAttributes = new BasicAttributes();
        userAttributes.put("objectClass", "top");
        userAttributes.put("objectClass", "posixAccount");
        userAttributes.put("objectClass", "Person");
        userAttributes.put("objectClass", "inetOrgPerson");
        userAttributes.put("givenName", newPerson.getFirst_name());
        userAttributes.put("sn", newPerson.getLast_name());
        userAttributes.put("uid", newPerson.getUsername());
        userAttributes.put("mobile", newPerson.getCell());
        userAttributes.put("mail", newPerson.getEmail());
        userAttributes.put("title", newPerson.getGender());
        userAttributes.put("userPassword", newPerson.getPassword());
        Name userDn = makeDN(newPerson.getFirst_name() + " "
            + newPerson.getLast_name());
        ldapTemplate.bind(userDn, null, userAttributes);
    }
}
```

Listing 5-11: Create User Account

The first thing on this class is the definition of springs' LDAP template properties which is annotated with springs' "Autowired" annotation which indicates that its value will be injected during runtime. The method that creates the account start by defining java naming directory attributes to collect all necessary attributes required to save an object on the directory server. It then later makes use of annotated LDAP template property to gain access to the directory server and bind the object. To bind an object means to create and save a new object identifiable with given a distinguished name known as "DN" (userDn).

```
<bean id="contextSource" class="org.springframework.ldap.core.support.LdapContextSource">
    <property name="url" value="ldap://localhost:389" />
    <property name="base" value="dc=myldap,dc=com" />
    <property name="userDn" value="cn=admin,dc=myldap,dc=com" />
    <property name="password" value="password" />
</bean>
<security:authentication-manager alias="authenticationManager">
<bean id="userSearch" />
<bean id="ldapAuthProvider" />
<bean id="ldapTemplate" class="org.springframework.ldap.core.LdapTemplate">
    <constructor-arg ref="contextSource" />
</bean>
<bean id="ldapUser" class="useraccount.soap.services.personImpl">
    <property name="ldapTemp" ref="ldapTemplate" />
</bean>
<bean id="personImpl" class="useraccount.soap.services.personImpl" />
```

Listing 5-12: Application Context - Context Source

The context source bean above defines the spring's configuration to get a connection from the directory server. The spring's LdapContextSource class needs to know at least four properties to successfully establish a usable connection. The ldapTemplate bean in Listing 5-12 is then



constructed with the first bean that establishes the connection. As indicated earlier the “personImpl” class has a spring LDAP template defined as property whose value is injected via a setter method as shown on the “ldapuser” bean as indicated.

### 5.3.2 Document Management System

This system is developed only to manage documents for users registered on the user account management system defined above. It is developed separately for that system just to more disparate systems or component that can be used to demonstrate system interoperability. The model in Listing 5-13 shows basic properties needed for each document to be saved.

```
@Entity
@XmlRootElement
public class FileInfo implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int fileId;
    private String fileName;
    private Date uploaded;
    private String uploadedBy;
    private String fileDir;
```

Listing 5-13: Document Data Model

Again the model is annotated with JPA annotations with an auto-generated identifier. To successfully upload a document on the server, file name and owner’s username are mandatory properties that are enforced on the client side.

```
@Stateless
@LocalBean
@WebService(endpointInterface = "docman.services.FileServiceRemote", portName = "SO
public class FileService implements FileServiceRemote, FileServiceLocal {
    @PersistenceContext(unitName = "DocMan")
    private EntityManager em;

    public FileService() {}

    @Override
    public boolean addFile(FileInfo file, String fileSource, InputStream ins) {
        FTPClient ftpClient = new FTPConnection().connect();
        InputStream ins = is;
        boolean done = false;
        if (ftpClient.isConnected()) {
            System.out.println("Connection successfully established...");
            if (ins != null) {
                try {
                    ftpClient.setFileType(FTP.BINARY_FILE_TYPE);
                    ftpClient.setFileTransferMode(FTP.BINARY_FILE_TYPE);
                    ftpClient.makeDirectory(file.getUploadedBy().replace(" ", "_"));
                    String filePath = file.getFilePath() + ".pdf";
                    file.setFilePath(filePath);
                    done = ftpClient.storeFile(file.getFilePath(), ins);
                    if (done) {
                        em.persist(file);
```

Listing 5-14: Uploading a Document

The application is implemented in Enterprise Java Beans (EJBs). As shown in Listing 5-14,

uploading a document is a two-step process that requires a persistence context (JPA) and connection to the underlying FTP server to keep the documents. The persistence context defines all the configurations and constraints to gain access to the data source in order to persist document details. The object that is responsible for persisting this JPA object is JAVA entity manager which gets a connection from an injected persistence context using JAVA “PersistenceContext” annotation.

However, before saving document details, the actual document must already be on a server. Apache FTP client is used in this project to communicate with the FTP server. The method in the listing takes in the document model, the source of the document and the input stream which is the content of the document. It then establishes a connection, prepares the client and creates the directory in the server for the coming document. If the document is successfully uploaded, the details are then saved in the database. The point of having a database is to avoid contacting the FTP server every now and then. In addition to that, FTP server may not be as flexible as a relational database in querying documents properties.

### 5.3.3 Maternity Management System

This is a very basic system that is used to capture diagnosis information for a woman during maternity process. The system has four models for the patient details, monthly check-ups, and labor and child details. The model for patient details has all properties that must be supplied during the registration of a patient. It is important to remember that this system is used to demonstrate interoperability. The patient details should be discoverable from user account management system defined above if the patient is registered. If the patient is not registered, a new account will be created. As patients attend monthly check-ups, the check-up data model is used to populate check-up records for the specific patient. The method to add new check-up records is shown in Listing 5-15.

```
@Override
public Checkup addCheckup(Checkup checkup) {
    Checkup result = null;
    if (checkup == null) {
        throw new NullPointerException("Checkup object not initialized");
    }
    try {
        result = checkupRepo.save(checkup);
    } catch (Exception e) {
```

Listing 5-15: Patient Monthly Check-up

The method takes in check-up model and passes it through to the spring repository interface for persistence. The labor data model is used when the patient is giving birth. This method has all properties that capture the whole process of labor including whether it was successful or not. If the labor was a success, the child model will be used to capture the child details such as gender, names, and weight (kilograms).

```
@RequestMapping(value = "maternity/checkup/add", method = RequestMethod.GET)
public ModelAndView addCheckup(ModelMap map, Checkup checkup, HttpServletRequest request) {
    String id = request.getParameter("maternity");
    map.addAttribute("maternity", checkup.getMartenalId());
    .....
    String delivery = request.getParameter("delivered");
    if (delivery.trim().equals("yes")) {
        Patient patient = matService.getMaternityById(Integer.parseInt(id));
        String labourstatus = request.getParameter("labourstatus");
        .....
        checkService.addCheckup(checkup);
        if (labourstatus.trim().equals("success")) {
            Child child = new Child();
            .....
            lab.setChild(chld);
        }
        labService.add(lab);
    } else {
        checkService.addCheckup(checkup);
    }
    .....
}
```

Listing 5-16: Maternity Process

Listing 5-16 show how all the defined models are used to complete the whole process of giving birth as explained above.

### 5.3.4 Birth Certificate System

The idea behind this system is to get details about any records of labor and newborns from an external system. As indicated earlier that this system assumes a particular case at home affairs office where officer process birth certificate applications. More details about getting all records from the external systems will be covered later on integration and interoperability section in this chapter. Listing 5-17 shows some of the operation that can be performed on this system regarding the processes of birth certificate application.

```

@Service
public class BEService implements IBEService {
    @Autowired
    private BCService service;

    public List<CollectionStatus> applications() {}

    public CollectionStatus checkStatus(int parentId) {}

    public SystemMessage createApplication(Labour labour) {}

    public List<BCertificate> approved() {}

    public SystemMessage BatchApplication(String jsonData) {}
}

```

Listing 5-17 Birth: Certificate Application Operations

When an application has been created, a number of operations can be performed on it. These includes tracking the status of the application, changing application status to complete and successful and listing all birth certificates that are ready for collection. Listing 5-18 shows how an application can be created.

```

public SystemMessage createCertificate(Labour labour) {
    SystemMessage message = new SystemMessage();
    BCertificate application = new BCertificate();
    try {
        String fullName = labour.getMaternal().getSurname() + " "
            + labour.getMaternal().getNames();
        application.setBcNumber(generateBCNumber(labour));
        application.setBcNumber(labour.getBirthNo());
        application.setBirthNumber("" + labour.getBirthNo());
        application.setChildNames(labour.getChild().getName());
        application.setSurname(labour.getChild().getSurname());
        application.setGender(labour.getChild().getGender());
        application.setParentFullNames(fullName);
        application.setMaternalId(labour.getMaternal().getPid() + "");
        application.setCollectReady(false);
        application.setCreationDate(new Date());
        BCertificate certificate = repo.save(application);
        message.setMessage("Completed :" + certificate.getBirthNumber());
    } catch (Exception e) {
        message.setMessage(MessageType.Exception);
        message.setMessage(e.getMessage());
    }
    return message;
}

```

Listing 5-18: New Application

The birth certificate model takes information from labor details and creates an application. The labor object that is received for an application also has both parent (patient) and child models. Again the repository used here to save the application makes use of the spring data JPA interface.

## 5.4 Web Services

Literature has indicated that one of the most appropriate and appealing approaches in developing applications that need to interact with another is a web service approach. This section provides details of web services implementation as per the scope of this research. It should be noted that

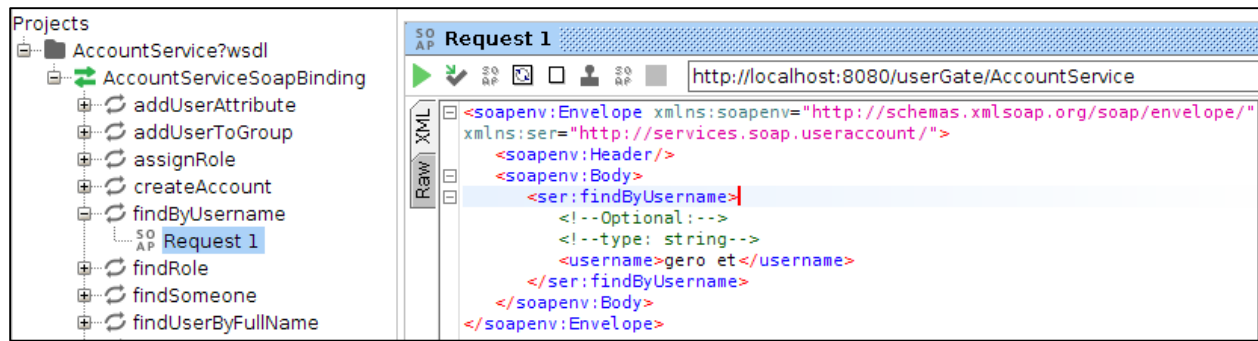
web services used in this research define entry points for sharing information with other components that need reuse of the same information. The first application/system that needs to share and expose its data to other external system is a user account management system as it needs to store and retrieve user information from the directory server for all other systems that need to use such information. SOAP web services were then implemented in this system to offer that functionality.

Most data models that were shown in the sections above were annotated with `@XmlElement` as to provide means for these models to be marshaled and transmitted over the network. WILDFLY application server comes with a full package of web service stack based on Apache CXF. This allows the developer to easily define basic web service only by providing relevant annotation on the relevant java classes. The requirement is that the class or interface that has methods to expose as web service be annotated with `@WebService` annotation. The second annotation that should be applied to the specific method is a `@WebMethod` annotation. If that method has parameters, it is ideal to use `@WebParam` annotation on all parameters so as to allow easy identification and understating of the parameters on any client that will be calling the web service. With these annotations properly applied, the deployment of the application will then provide a brief information about the web service created on the console as depicted in Listing 5-19.

```
20:49:10,144 INFO [org.jboss.ws.cxf.metadata] (MSC service thread 1-1)
address=http://localhost:8080/userGate/AccountService
implementor=useraccount.soap.services.personImpl
serviceName={http://services.soap.useraccount/}AccountService
portName={http://services.soap.useraccount/}AccountServicePort
annotationWsdllLocation=null
wsdlLocationOverride=null
mtomEnabled=false
```

Listing 5-19: WILDFLY Console - Web Service Deployment

The console shows that a SOAP web service has been created and it can be accessed from the indicated address. There is not much information about this web service, for more detailed information, what is known as web service definition language can be consulted by simple suffixing the web service address with “?wsdl”. Listing 5-20 shows some of the operations that are available on this web service and how the service can be invoked and tested.



Listing 5-20 SOAP Service - SOAP UI

Each operation defined as web service method has a corresponding response object. As shown in the figure on Listing 5-20 each web service method/operation is represented as an XML object called SOAP envelope. The content or the message that is being communicated to and from the service is contained in the soap body. As an example in the figure above, the literal string username that is an input parameter in that service operation is defined inside the soap body. Most of the system components in this research employ SOAP as web service approach and follow the same step as explained above.

Explained below is the implementation of the REST web service as it is also used on some of the systems. To begin with, the maternity system used in this research exposes some of its operations as rest web services. The implementation is a little different as it directly implements Apache CXF rather than relying on WILDLFY (RestEasy) implementation of CXF. However, it still follows the java core rest web service with annotations as shown in Listing 5-21.

```
@Path("/")
@Service
public interface IWclinic {

    @GET
    @Path("/checkup/list/{patientid}")
    @Consumes("application/json")
    @Produces("application/json")
    public @ResponseBody List<Checkup> checkuplist(@PathParam("patientid") String patientId);

    @GET
    @Path("/patient/children/{patientid}")
    @Consumes("application/json")
    @Produces("application/json")
    public @ResponseBody List<Child> children(@PathParam("patientid") String patientId);

    @GET
    @Path("/patients")
    @Consumes("application/json")
    @Produces("application/json")
    public @ResponseBody List<Patient> patients();
}
```

Listing 5-21: Service Interface - Resource Class

The path annotation defines the context or root path from which the web service will be accessed. The methods or operation defined in the interface and are to be exposed as rest service operations need to be decorated to indicate HTTP request method supported for that specific method. In Listing 5-21, the @GET annotation on the first method shows that the method supports HTTP GETrequest. The path annotation on the method or operation also indicates the address to access that specific method as the web resource. It is also ideal to specify the type of data the method accepts, the two annotations @Consumes and @Produces define the type of data the method accepts and the type of data the application produces, respectively. Another key annotation used here is the @PathParam which helps get data provided as path parameters on the resource address. In other words this annotation is used to bind path values to parameters of a particular method. The next important step is to register the class that implements this interface as JAXRS bean that exposes the rest service.

```
<jaxrs:server id="service"
  address="/rest"
  serviceClass="birthtech.services.rest.WClinic">
</jaxrs:server>
```

Listing 5-22: REST Service Server Configuration

Listing 5-22 shows how spring bean configuration file is used define a server side bean that exposes a service class as a RESTfull web services. The jaxrs:server component needs to be configured with at least an address and a service class. The service class needs to have an interface with necessary annotations that defines REST resources as discussed above. The next step now is to register this configuration on the web.xml and define a servlet that will listen and handle web calls. The web.xml defined on Listing 5-23 will registerall configuration files on the application context. The CXFServlet configured will be responsible for managing CXF REST service calls that are directed to “/services/\*” URL pattern.

```

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    classpath:META-INF/applicationContext.xml
    classpath:META-INF/cxf-context.xml
    classpath:META-INF/cxf-rest.xml
  </param-value>
</context-param>
<servlet>
  <servlet-name>CXFServlet</servlet-name>
  <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>CXFServlet</servlet-name>
  <url-pattern>/services/*</url-pattern>
</servlet-mapping>

```

Listing 5-23: CXF Servlet

The next step at this point is to deploy the application on a supported application server which in this case is JBOSS WILDFLY. A web application definition language (WADL) can give a very clear understanding of what has been implemented so far with all the configuration above. The full address of the web service is composed of root context of the application, the mapping or URL pattern of the CXF servlet and lastly, the address part defined on the jaxrs:server component. The details are shown in Listing 5-24.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2009/02" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <grammars>...</grammars>
  <resources base="http://localhost:8080/BirthTech/services/rest">
    <resource path="/">
      <resource path="checkup/list/{patientid}">
        <param name="patientid" style="template" type="xs:string"/>
        <method name="GET">...</method>
      </resource>
    </resource>
  </resources>
</application>

```

Listing 5-24: REST Service WADL

The WADL refers to the full address described above as a resource base. It is noticeable at this point that REST embraces the concept of a resource for each object or class and its operations or methods as can be seen above that the path “/” that was given to service interface is a resource on its own while at the same time the path “checkup/list/{parentid}” given to a method that lists checkups for an individual is also referred to as a resource that takes a literal string as a parameter “param” and accepts an HTTP GET request. The set of configurations defined in this section is enough for the definition of a basic REST service.



## 5.5 Integration and Interoperability

So far this chapter has covered the implementation of stand-alone applications implemented as the requirement of this research. A number of technologies have been introduced and explained how they are used in the context of this research. It is now time to look at the last and very key concepts of this research which are system integration and interoperability. This section will focus more on enabling these standalone disparate applications to communicate and share information. **Error! Reference source not found.** below shows at a very high level the type of interaction required at the end.

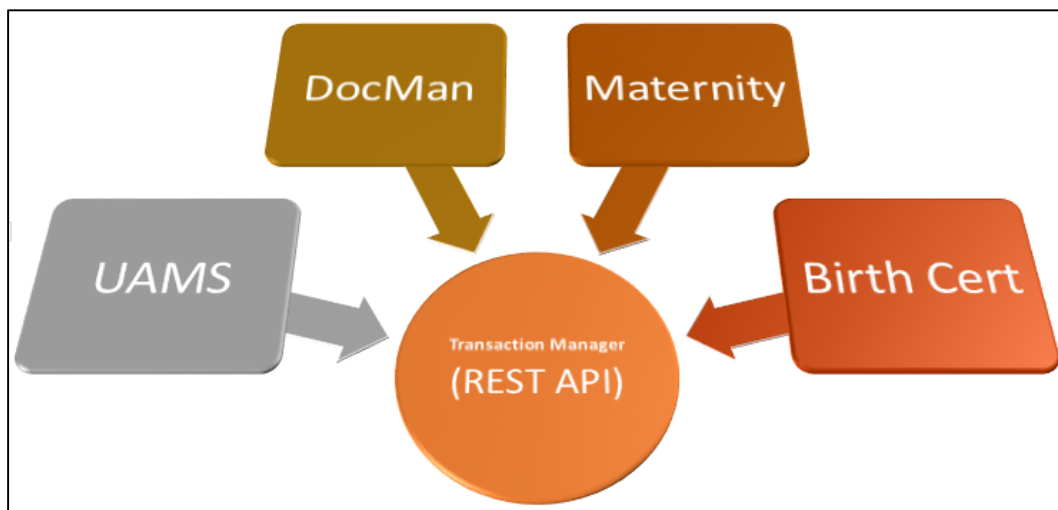


Figure 5-1: Component Interaction

The four components that have been discussed so far are expected to expose an entry point that allows other external systems to query and use their information. As depicted above, each of the components or systems is able to connect to the transaction manager component. It is the responsibility of the transaction manager component to accept requests from various clients then create messages and dispatch them to the appropriate component. The transaction manager must then take the response from the external disparate systems and present it to the client. That created the initial request. Sometimes the transaction manager is going to be fed with requests that need information from more than one stand-alone application, it is again the responsibility of the transaction manager to handle such kind of requests and give the client an appropriate response.

Although the transaction manager is REST API, it is capable of interacting with applications of various technologies such as other REST services, SOAP services, and JMS services. It is now clear that the transaction manager gives a layer of abstraction that allows clients not to

communicate with standalone applications directly but rather provide a channel that can integrate information from all standalone applications. Clients will now access various kinds of information from a single source through this REST API. It is also important to mention that the communication among all these different components is achieved by means of message routing using Apache Camel as a system integration framework. Before diving into details of routing implementation with apache camel, it is ideal to understand the prerequisites that will allow a smooth exchange and reusability of information on the participating endpoints.

Each component that is designed to share its information with other components or systems should take into count the following concerns. It has to expose an entry point to share that information by means of either SOAP, messaging (e.g JMS) or REST service. The next important concern is the sharing/exposure of data models. This could be a simple task in the SOAP service since artifacts/data models can easily be generated from the WSDL. In the case of REST service, the details of JSON providers must also be defined so as to make sure that client applications use the same or related JSON provider which will be able to understand and transforms (serialize and deserialize) the data without distortions. The client application should have knowledge of the type of data each resource in a REST service consumes and produces. WADL will provide basic information, however, should a service not provide support to expose WADL, it must have other means to provide such information. Although SOAP service is mostly used with HTTP, it must also be indicated to the client which protocol is used for data exchange. These are some key considerations that need to be properly defined when designing applications that communicate with other applications.

To discuss how this research achieved the integration of all the systems used in this research, it must be understood that the approach followed is that of an open service gateway initiative (OSGI). The approach provides a flexible approach for defining system modules that are flexible and reusable by other various systems. This modularity is very key in the context of this research as it can deal with specific functionalities that are needed by various components. The first implementation of OSGI in this context is used for customizing user profile from two systems namely: UAMS and DocMan. An OSGI bundle defined to handle this feature encloses apache camel routing capabilities. The good thing about OSGI and its implementation in this research is that it does not need too many configurations and it allows the developer to focus mainly on the relevant features. As results Listing 5-25 shows minimal steps needed to define an OSGI bundle.

The most important part in the figure below is to define project packing on the POM as a bundle. Packing a project as a bundle will produce jar file with all contents of the project including a MANIFEST file that provides more details about the bundle.

Apache Felix was used as a plugin in this project to handle the generation of bundles. The plugin needs to be configured with a bundle symbolic name which is used uniquely identify the bundle. All the packages that should be included as dependencies of the bundle also need to be defined in the plugin. All this information will be written to the MANIFEST file that the plugin will generate.

```
<artifactId>infosource-upservice</artifactId>
<packaging>bundle</packaging>
<version>1.0.0-SNAPSHOT</version>
<name>infosource-upservice</name>
<properties>
</properties>
<repositories>
</repositories>
<dependencies>

<build>
  <defaultGoal>install</defaultGoal>
  <plugins>
    <plugin>
    </plugin>
    <plugin>
    <!-- OSGI Bundle Manifest definition -->
    <plugin>
      <groupId>org.apache.felix</groupId>
      <artifactId>maven-bundle-plugin</artifactId>
      <version>2.3.7</version>
      <extensions>true</extensions>
      <configuration>
        <instructions>
          <Bundle-SymbolicName>Infosource-upservice</Bundle-SymbolicName>
          <Import-Package>*, !com.google.gson</Import-Package>
          <Export-Package>birthcertificate.ews.*</Export-Package>
          <Embed-Dependency>gson</Embed-Dependency>
        </instructions>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Listing 5-25: OSGI Bundle Definition

In this case, all packages except for Google gson package will be included in the bundle. Having successfully defined the necessary information for the MANIFEST file, when maven install goal is run, the plugin will be invoked and it will attempt to generate the bundle. It is now time to understand the logic inside the bundle. As indicated earlier that the bundle encloses apache camel routing capabilities, the first requirement for such routing is the definition of service endpoints.

```

<camelContext xmlns="http://camel.apache.org/schema/blueprint">
  <routeBuilder ref="routeBuilder" />
</camelContext>
<cxfrsClient id="MartEndpoint">
<cxfcxfEndpoint id="BCEndpoint">
  <!-- Doc Man -->
<cxfcxfEndpoint id="DocMan"
  address="http://localhost:8080/DocMan/FileService/FileService"
  wsdlURL="http://localhost:8080/DocMan/FileService/FileService?wsdl"
  serviceClass="docman.services.FileServiceRemote"
  serviceName="ws:FileService"
  endpointName="ws:SOAPFileService"
  xmlns:ws="http://services.docman/">
</cxfcxfEndpoint>
  <!-- User Account Manager System -->
<cxfcxfEndpoint id="useraccount"
  address="http://localhost:8080/userGate/AccountService"
  wsdlURL="http://localhost:8080/userGate/AccountService?wsdl"
  serviceName="ws:AccountService"
  endpointName="ws:AccountServicePort"
  serviceClass="useraccount.soap.services.PersonInterface"
  xmlns:ws="http://services.soap.useraccount/">
  <cxfp:properties>
    <entry key="dataFormat" value="POJO"></entry>
  </cxfp:properties>
</cxfcxfEndpoint>

```

Listing 5-26: Endpoint Definition

The two endpoints shown on Listing 5-26 are be used in apache camel routes in a route builder class. It is noticeable that these are disparate applications that both implement SOAP web service. Defining a SOAP endpoint requires a number of key elements of the service although some are only necessary to define for specific data exchange format. As an example, the service class is necessary when using a POJO plain old java object (POJO). Other elements such as service name and service endpoint name can reference through XML namespace the definition in the service contract. Listing 5-27 below shows how these endpoints are used to define routes on a builder class that is registered in the camel context.

```

public void configure() throws Exception {
  from("vm:userProfile").choice()
    .when(header("profileMode").isEqualTo("full"))
      .process(new SetBodyProcessor())
      .to("direct:subRoutes")
    .when(header("profileMode").isEqualTo("docs"))
      .process(new DocManProcessor())
      .to("direct:DocManEndpoint")
    .when(header("profileMode").isEqualTo("basic"))
      .process(new UAServiceProcessor())
      .to("direct:UAEndpoint")

```

Listing 5-27: Routes

The routes defined in the figure above are created using apache camel JAVA DSL as opposed to spring DSL. Apache camel has a number of components to allow routing to and from various endpoints for various scenarios. As the architecture used in this research follows a component or modular based development, it is important to make sure that component can easily access

information on other modules especially those running on the same virtual machine. In this case, the route above defines the backend service that provides access to user profiles. The request for such information could come from a number of other disparate modules with the same machine. The apache camel “VM” routing component makes it possible for multiple apache camel contexts running in the same java virtual machine to discover each other’s endpoints. This means endpoints can asynchronously exchange messages with other endpoints running on separate thread pools. Apache camel allows content based routing which makes it easy to determine the endpoint to send a request to base on the actual message body or header being communicated.

This type of routing is used above to understand the requested filtering for the user profile. If for example, the client wants full records of a particular user, a “profileMode” header variable must be assigned a value of “full”. This means that each message received from “vm:userProfile” can be directed to one of the two options available or both of them. In a case where the client wants a complete record of a user profile, the request message will be forwarded to both endpoints referred to as “direct:subRoutes” on the listing above. By default, apache camel applies pipeline routing where the request message is manipulated and forwarded to the next endpoint in the pipeline.

Listing 5-28 shows how multicast routing is used in this project to send the same request to more than one endpoint that holds user records. The direct component will synchronously invoke an existing route in the context.

```
from("direct:subRoutes").multicast()
    .aggregationStrategy(new AggregationStrategy() {
        public Exchange aggregate(Exchange oldExchange,[]
    }).to("direct:DocManEndpoint", "direct:UAEndpoint");

from("direct:DocManEndpoint").setHeader("operationName")
    .simple("getFilesByOwner").to("cxf:bean:DocMan").marshal()
    .json();
from("direct:UAEndpoint").setHeader("operationName")
    .simple("findByUsername").to("cxf:bean:UserGate").marshal()
    .json();
```

Listing 5-28: Multicast Routing

As noticeable on the listing, multicast requires an aggregation strategy that defines how the responses from the list of consumers should be formatted and put together. The list of endpoints that are targeted for this unchanged requests should then be placed on the “to” method. The two endpoints in this case that will receive the request are both defined using the “direct” component. This so as to allow the definition of these endpoints individually in a manner that makes them

reusable and easy to define further configurations for each. This means that instead of defining “cxf:bean:DocMan” for Document Management System and “cxf:bean:UserGate” for User Account Management System directly in the multicast endpoint list, these two endpoints are defined separately to allow further configurations such as header modification and marshalling features while also reusable for other needs. The “cxf” component used above provides the integration with apache CXF web services. This means that the endpoint used with this component is an Apache CXF web service. As indicated earlier apache camel can easily set header and body values directly on the routes, for more complex situations that deals with the manipulation of the message being communicated, apache camel allows the definition of a processor interface. As an example, each condition (“when”) on the route definition above there is a processor that processes the body content before it is sent to the desired endpoints. This could easily be achieved using the direct “setBody” operation, but because the targeted endpoint is a SOAP web services that is configured to access data in POJO mode which understands body content as apache CXF MessageContentList. Listing 5-29 shows how a processor is used to enclose the request message body in a MessageContentList.

```
public class SetBodyProcessor implements Processor {
    @Override
    public void process(Exchange exchange) throws Exception {
        MessageContentsList msgContentList = new MessageContentsList();
        msgContentList.add(exchange.getIn().getBody(String.class));
        exchange.getOut().setBody(msgContentList);
    }
}
```

Listing 5-29: Apache Camel Processor

The exchange object that is accepted as a parameter in the process method is sort of a container that encapsulates both header and body of the message being communicated. This also provides details of an outgoing (Out) message and incoming (In) messages. The request in this process is extracted as a string object from the incoming body (getIn()). The extracted string is then added to the message content list object as required by CXF (SOAP) web service endpoint. The message content list is then set as the body of an outgoing (getOut()) message. It is the modified exchange that will be forwarded to the desired endpoint.

Another useful feature that is used in the definition of user profile routes is the marshaling of responses to JSON object. Although some services in this research are defined using SOAP web

services, the plan was to have an integrated client exposed as a REST API. That is why the top level processing needs the messages to be in JSON format that will be easily managed by the rest service. It is important to mention that camel context that comprises the routes and configuration above is then packaged in a separate OSGI bundle and deployed on an OSGI container separately. At this point, one can develop an interest in the implementation of the client that sends requests that will be handled by routes defined above. Listing 5-30 shows routes defined in camel context deployed as a separate OSGI bundle.

```
<cxfrs:rsServer id="profileclient" address="/upservice"
  serviceClass="infosource.upsclient.rest.UPService"
  loggingFeatureEnabled="true">
  <cxfrs:providers>
    <ref component-id="jsonProvider" />
    <ref component-id="cors" />
  </cxfrs:providers>
</cxfrs:rsServer>

<camel:camelContext id="camel">
  <camel:route>
    <camel:from uri="cxfrs:bean:profileclient" />
    <camel:setHeader headerName="profileMode">
      <camel:simple>${header.operationName}</camel:simple>
    </camel:setHeader>
    <camel:marshal>
      <camel:string />
    </camel:marshal>
    <camel:to uri="vm:userProfile"></camel:to>
  </camel:route>
</camel:camelContext>
```

Listing 5-30: User Profile Client Routes

The first part on listing above is the definition the endpoint bean that exposes a REST service to create user profile requests. This bean also needs to be configured with a service class that defines all rest methods or resources for specific profile mode as indicated earlier. The service class in this instance does not need logic implementation for its methods or operation. This means that the methods must be well defined without logic implementation and can return null. This is because apache camel will take over and handle the request directed to those REST operations. The routes on the listing are now defined using spring DSL inside camel context. As depicted the route starts off by consuming request message sent to the endpoint described earlier on. The “cxfrs” component is used to enable features for consuming JAX-RS web services. The route modifies the exchange by defining a header variable that holds profile mode with its value being the web service operation name. This profile mode header variable is the one that is interrogated on the content-based routing earlier. Marshalling is used to make sure that the message body is in string format as the processor will attempt to extract it as a string. Finally, the request message is sent to

“vm:userProfile” endpoint. This endpoint is defined in another camel context in a separate OSGI bundle as discussed. The response will then be presented to the client in a JSON format. With these two OSGI bundles prepared this way, it is now safe to say two disparate system modules are able to exchange information during runtime.

Besides these two OSGI bundles, there are other bundles that define other services used in this work. These include a bundle for the woman’ clinic system (maternity management system) and the birth certificate system. As the implementation of these two systems has been presented, the bundles only define routes that consume services exposed by the systems. As each system exposes some of its operation using web services, the bundles are defined separately to define messaging around the exposed web resources and are deployed separately on OSGI container. This means each application runs on an application server and can be used internally from that application server while some of its functionalities are exposed on OSGI bundle. It is now time to look at how these other applications can use those bundles to share and exchange information.

It was said earlier on that every time a baby is born at any health center, the details of labor process including those of a mother and a baby should be communicated to the birth certificate system (which is referred to as a home affairs component). A separate OSGI bundle that embraces apache camel routing features was created separately to define logic that constantly checks if there are new records on woman’s clinic system. If there is a new record, camel routes defined in this bundle should then forward these records to the birth certificate system. These should not need human intervention but should happen automatically. The officer responsible for processing birth certificates should be able to see these records and take it from there. The route contained in this OSGI is shown in Listing 5-31.

```
public void configure() throws Exception {
    from("timer:myTime?period=50000")
        .process(new infosource.messaging.rest.PrepareRestProcessor())
        .to("cxfrs:bean:rsClient").marshal().json()
        .process(new PrepareSoapProcessor())
        .to("cxf:bean:soapServiceHA");
}
```

Listing 5-31: Automatic Record Exchange

The route makes use of timer component that allows to constantly invoke the entire route to check if the new records. Since this an automatic or timer based route, it is the responsibility of a processor to create a new request that will be forwarded to the endpoint. The exchange created by the processor is then forwarded to “cxfrs” composed based endpoint which means it’s a REST



service. Woman's clinic system exposes its operations using REST web services and its response will be in a JSON format. Because new records if exists will be forwarded to a SOAP service, the processor shown below is used to manipulate the JSON objected to the created SOAP service compliant message body.

```
public void process(Exchange exchange) throws Exception {  
    Message in = exchange.getIn();  
    String jsnString = in.getBody(String.class);  
    MessageContentsList messageContentList = new MessageContentsList();  
    JSONObject jobject = new JSONObject(jsnString).getJSONObject("list");  
    String str = jobject.getString("infosource.messaging.models.Labour");  
    messageContentList.add(str);  
    exchange.getOut().setHeader("operationName", "BatchApplication");  
    exchange.getOut().setBody(messageContentList);  
}
```

Listing 5-32: Preparing Message Content List for SOAP

The processor on Listing 5-32 retrieves the received message as a JSON string which it uses to create a JSON object. A JSON-formatted string that represents a labor process data model as explained earlier is again retrieved from the JSON object and added to the message content list to be channeled over SOAP web service. When this string reaches the endpoint it will be converted again to a JSON objected which will be used to reconstruct a labor process data model that will be used to keep the record as a new application for a birth certificate. It should be noted that labor process data model is shared between these two applications.

## 5.6 Security Implementation

The introduction and background information on ICT4D and e-government presented in this work cited that security is one of the key areas that need solid focus. Literature review presented on chapter two also made mention of how security has been a concern following the rapid advancement and adoption of sophisticated telecommunication systems (Andress, 2014; Rafique and Humayun, 2015). Although security is a broad area of research itself which is the not the focus of this work, it is, however, ideal to put it a top level priority in the design and implementation of ICT systems of this nature as stipulated in the literature review. This work has therefore implemented minimal security features that embrace the CIA triad principle as explained in chapter two in order to provide a secure platform that offers confidentiality, integrity, and availability while maintaining the security requirement in e-government system design and development. All systems and components used in this research rely on the HTTP protocol to offer the intended

communications and transactions.

HTTP is one protocol that is mostly used by a vast number of developers in the deployment of web services and applications (Li, Bi and Wang, 2013; Che and Tuo, 2016). One can define HTTP as TCP/IP stateless protocol that works as the request-response model (Che and Tuo, 2016). In order to ensure that web services and applications deployed over HTTP are secured, transport layer security needs to be implemented. TLS/SSL protocol can be used to secure communications and transactions over HTTP. TLS (Transport Layer Security) as the successor of the SSL(Secure Sockets Layer) protocol can be referred to as cryptographic protocol that offers confidentiality, integrity, and availability among other security features using asymmetric cryptography (Jeelani, 2013; Husák *et al.*, 2015). Discussed below are the details of how TLS security has been applied to this research to secure HTTP communication resulting in the use of HTTPS. It should be noted again that this research implements the minimal requirement of security as needed in the implementation of ICT systems this research is concerned about.

All web applications and services used in this work are deployed on JBoss WILDLFY application server and are accessed HTTP protocol. The first requirement in providing a secure channel for accessing this application by using HTTPS is to define security keys as per the requirement of TLS/SSL. In order to generate the necessary keys and certificates to configure TLS java Keytool and OpenSSL can be used.

As these keys and certificates are to be used by both server and client as they need to perform a transaction, it is important that the certificates are signed by a trusted Certificate Authority (CA). It is, however, possible to use a self-signed certificate although not encouraged in a production and public environment. For the purpose of this research, a self-signed certificate generated and used. Shown in Listing 5-33 are some of the java keytool commands.

```
lizo@lizo-Aspire-R3-131T:~/certs$ keytool -genkey -alias wildfly -keyalg RSA -k  
eysize 2048 -validity 365 -keypass secret -keystore serverkeystore.jks -storepas  
s secret
```

Listing 5-33: Generating Keys and Certificates

This command uses RSA algorithm to generate a 2048-bit key pair valid for 365 days. The keytool will put the key and the generated self-signed in a keystore. It is possible to export and import certificates from and into this keystore.

In order to use the generated self-signed certificate on WILDFLY application server, an SSL

security realm and HTTPS listener need to be defined in the application server's configuration file. Listing 5-34 below shows the definition of SSL realm.

```
<security-realm name="SslRealm">
  <server-identities>
    <ssl>
      <keystore path="/home/lizo/certs/serverkeystore.jks" keystore-password="secret"/>
    </ssl>
  </server-identities>
</security-realm>
.....
<http-listener name="default" socket-binding="http"/>
<https-listener name="default-ssl" socket-binding="https" security-realm="SslRealm"/>
```

Listing 5-34: SSL Realm

The realm needs to be configured with the keystore path and the password that it can use to get access to the key store. The listener is defined by setting the security realm for the HTTPS socket binding.

Another very crucial part of security that is implemented on all systems on this work is the authentication and authorization mechanism. This mechanism is based on a role-based access control model implemented using spring security and LDAP. Shown in Listing 5-35 the first step needed to secure URL patterns based on the defined roles.

```
<security:http auto-config="true">
  <security:session-management
    invalid-session-url="/spring_security_login">
  </security:session-management>
  <security:intercept-url pattern="/web/**"
    access="ROLE_User, ROLE_Admin" />
  <security:intercept-url pattern="/admin/**"
    access="ROLE_Admin, ROLE_Master" />
  <security:logout logout-url="/web/logout" />
</security:http>

<security:authentication-manager alias="authenticationManager">
  <security:authentication-provider
    ref="ldapAuthProvider" />
</security:authentication-manager>
```

Listing 5-35: Securing URL Pattern

This configuration defines two base URL patterns (“/web/\*\*” and “/admin/\*\*”) that need to be secure. The first one can be accessed by a user who has at least a “User” role while the second can be accessed by a user with “Admin” role. By default, deploying an application with only this configuration result in spring security intercepting all URL matching the pattern above for authentication. The second part of the configuration is the definition of the authentication-manager which needs to be given then chosen authentication provider. In this work LDAP is an authentication provider of choice. The authentication provider, in this case, relies on a number of beans in order to work properly. Some of these beans are shown in Listing 5-36

```

<bean id="contextSource" class="org.springframework.ldap.core.support.LdapContextSource">
  <property name="url" value="ldap://localhost:389" />
  <property name="base" value="dc=myldap,dc=com" />
  <property name="userDn" value="cn=admin,dc=myldap,dc=com" />
  <property name="password" value="password" />
</bean>
<bean id="userSearch"
  class="org.springframework.security.ldap.search.FilterBasedLdapUserSearch">
  <constructor-arg index="0" value="" />
  <constructor-arg index="1" value="(uid={0})" />
  <constructor-arg index="2" ref="contextSource" />
</bean>

```

Listing 5-36: LDAP Context Source

The context source bean is used to define the connection to the underlying LDAP server that has user details and roles. The “userSearch” bean defines the LDAP attributes from which necessary user information should be extracted.

```

<bean id="ldapAuthProvider"
  class="org.springframework.security.ldap.authentication.LdapAuthenticationProvider">
  <constructor-arg>
    <bean
      class="org.springframework.security.ldap.authentication.BindAuthenticator">
      <constructor-arg ref="contextSource" />
      <property name="userSearch" ref="userSearch" />
    </bean>
  </constructor-arg>
  <constructor-arg>
    <bean class="customapp.security.authoritiesPopulator" />
  </constructor-arg>
</bean>

```

Listing 5-37 Authentication Provider

Listing 5-37 shows the provider bean which also has a bean that loads user roles. In spring security terms, the bean that loads user roles is known as “authority’s populator”. The logic of this bean is such that, after the user is authenticated, it used the username or Id to search all matching LDAP attributes that represent roles.

## 5.7 Conclusion

This chapter presented a detailed implementation of all systems and components as prototypes to demonstrate the key concepts of this research. The chapter started by presenting the implementation details of the customizable prototype for the cataloging stage of electronic government. All other systems that were presented after that were meant to showcase the implementation of system integration. The systems adopted Apache CXF web services to expose channels to communicate and share information with other systems. Apache camel was used for system integration implementation to define a seamless interaction and exchange of information across all four component discussed. The implementation of security features to provide a secure platform to consume services was also presented.

## **Chapter Six: System Testing**

### **6.1 Introduction**

This chapter illustrates system testing methods for all components of the different systems presented so far in this research. Unit tests were employed to test and maintain the desired functionalities on specific components and units of the systems throughout the development and the life cycle of the system components. Integration tests were used to ensure that all layers within each system and all other participating components could seamlessly work together as desired. It should be noted that as the actual service implementation classes depended on other services or data sometimes, such dependencies are mocked in order to pay more attention to the specific functionality or unit as per the nature of unit testing. Within the context integration test, as this research has produced, a REST API that acts as a client to most participating systems and component in striving to provide one access point for various transactions, it is then important to make sure that all service exposed by the REST API are tested accordingly.

### **6.2 Unit Testing**

Unit testing is used in this search to ensure that the functionalities for each system component maintained the intended behavior throughout the life cycle of the component. To begin with, this research started off by investigating an approach to reuse an existing customizable system for dissemination of information across various department and organizations. This resulted in a system herewith referred to as “Custom App”. As indicated earlier that this application is implemented using spring framework, it is composed of a spring controller, spring data JPA interface and actual service implementation class in between. These are treated by spring framework as beans that make use of dependency injection to call each other and other beans. Some of the functionalities needed from this system include the ability to create a domain, get a list of all domains and find a domain by name. Since the actual logic implementation of this system requires a repository interface as a bean that performs database transaction, that interface, and all other dependencies are not of interest at this level hence we mock (fake) them out and perform unit testing.

```

@RunWith(MockitoJUnitRunner.class)
public class TestDomainServices {
    @Mock
    private DBCreator creator;
    @Mock
    private DomainRepo repo;
    @InjectMocks
    private DomainDAOImpl SUT;
    @Rule
    public final ExpectedException exception = ExpectedException.none();

    @Before
    public void BeforeTest() {
        MockitoAnnotations.initMocks(this);
        SystemMessage sysMsg = new SystemMessage();
        sysMsg.setSuccess(true);
        when(repo.findAll()).thenReturn(TestData.getListOfDomain());
        when(creator.create(any(Domain.class))).thenReturn(sysMsg);
    }
}

```

Listing 6-1: Unit Test Setup

Shown in Listing 5-1 is the setup of the test fixture or class with a more clear insight of how dependencies that are injected into the system under test (SUT) are mocked out. It is, however, important to tell these mocked dependencies what data should be returned when a certain operation gets invoked. Again in the body of the specific test, it must be verified that really a certain operation was invoked with a specific input. When working with spring framework, it can be very cumbersome to mock multiple beans that are injected on service that is being tested. The use of `@Mock` annotation on all dependencies for SUT and the use of `@InjectMocks` for actual SUT provide a clear mechanism to overcome that part. The test setup method annotated with `@Before` annotation gets called for each test in the test class or fixture. This method prepares the mocks by initializing them and specifying which data to use. Mockito does provide a mechanism to match the SUT processing output with the expected output through Matchers. The listing below shows one of the tests that use the configuration discussed above.

```

@Test
public void should_be_able_to_get_a_list_of_domains() {
    List<Domain> result = SUT.getAllDomains();
    int expectedId = TestData.getListOfDomain().get(0).getDomainId();
    int actualId = result.get(0).getDomainId();
    verify(repo, only()).findAll();
    assertNotNull(result);
    assertEquals(actualId, expectedId);
    assertEquals(result.size(), TestData.getListOfDomain().size());
}

```

Listing 6-2: List Domain Test

Listing 6-2 gives a more clear view of how most unit tests for the domain service are written. In essence, the SUT is used to call its method that gets a list of all domains from a repository. Since the repository is mocked out and given output data, in this case, the interest is in the ability of SUT

to retrieve the data and pass it to calling client. It is then verified that through mockito verifications whether or not a relevant repository method to retrieve data was called. If the repository was called, SUT is further tested by JUnit assertion that it really did return something. The last two assertions verify that the data returned by SUT is actually the data that repository was instructed to return and it's not altered. Another functionality offered by this system is the ability to find a domain by name.

```
@Test
public void given_null_checkup_should_throw_nullpointer_exception() {
    Checkup check = null;

    exception.expect(NullPointerException.class);
    exception.expectMessage("Checkup object not initialized");
    SUT.addCheckup(check);
}
```

Listing 6-3: JUnit Rule - Expect Exception

Listing 6-3 shows other scenarios that can be tested for a specific functionality. In the case of maternity system defined this research, the system should be able to detect that the given input is not in the correct state to complete the required action. JUnit provides rules that can be used to expect exceptions when certain kind of input is fed into a specific unit of code. The exception variable declared in the test method is annotated with JUnit's `@Rule` annotation. Inside the test method, all the expectation needs to be specified before SUT is executed. Adding a null or uninitialized check-up object should result in SUT throwing a null pointer exception with the specified message failing which would mean that the "addCheckup" functionality does behave in an intended manner.

### 6.3 Integration Testing

The first thing to note from Listing 6-4 is that the test class is annotated with spring's context configuration. The context configuration file, however, is tweaked a bit to allow testing scenarios.

```

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "classpath:META-INF/applicationContext.test.xml" })
public class TestDomainRepository {

    @Autowired
    private DomainJDBC jdbc;
    @Autowired
    private DomainRepo SUT;
    TestDataManager TestDataManager;
    final String appContextPath = "META-INF/applicationContext.test.xml";
    private Domain domain;

    @Before
    public void setUp() throws Exception {
        TestDataManager = new TestDataManager(appContextPath);
    }
}

```

Listing 6-4: Integration Test Setup

Spring's auto wiring support is used to inject services as registered on the context configuration file. The data that is used and produced by this test class is managed by the "TestDataManager" class that is configured in the testing configuration file to point to a testing database while importing all other beans from the main application context. It might be clear at this point that this kind of test configuration and setup is preparation for integration testing. To begin this integration testing, it best to make sure that JPA interfaces work as needed and are able to perform database transactions with the defined JPA entities.

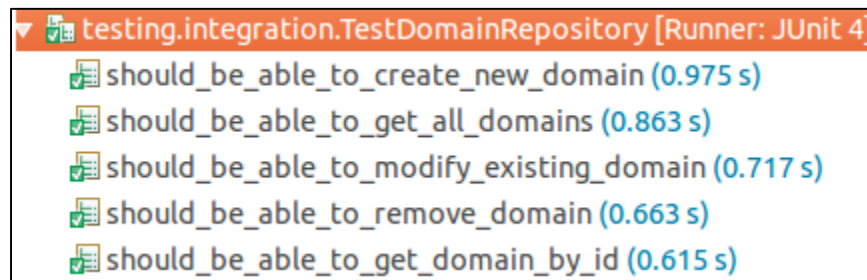


Figure 6-1: JUnit Window

Figure 6-1 shows a JUnit window with passing integration tests that focus on some of the CRUD operations offered by the JPA interface. The naming of the tests is very important in the sense that, every time the test class is visited it gives a clear indication of what has been tested. Understanding the test fixture configuration described above and having the JPA interface working as the one for domain services described above could indicate the readiness to perform integration test for actual logic implementation.

Another system that is used for the purpose of this research is a maternity management system as described in chapter 3 and 4. It follows spring setup and configurations as previously demonstrated with layers that include data access layer provided by spring data JPA. The actual service



implementation then injects the JPA interface as bean to perform transactions. Testing at this level needs service logic bean injected on the test fixture. Listing 6-5 show a very basic integration test.

```
@Autowired
private PatientService patientService;

@Rule
public final ExpectedException exception = ExpectedException.none();

@Test
public void given_null_patient_object_should_throw_nullpointer_exception() {
    Patient patient = null;
    exception.expect(NullPointerException.class);
    patientService.registerPatient(patient);
}

@Test
public void given_valid_patient_object_should_create_a_new_record() {
    Patient patient = TestData.GetPatient();
    boolean created = patientService.registerPatient(patient);
    assertTrue(created);
}
```

Listing 6-5: Integration Test

The logic implementation class for patient service as accessed through spring's auto wiring support is required to validate any patient object such that it is able to detect null pointers to the actual object or its properties. Through JUnit rules it is possible to listen as the targeted method is executed with an invalid object. The test fails if the expected exception is not thrown. The second test method passes a valid object to SUT and expects it to return true indicating creation success. Having tests of this nature truly passing to indicate the validity and readiness of all the layers involved so far means that, the service can now be called by a client or another service to perform a certain transaction. The calling service or layer then introduces another part of the hierarchy that needs integration testing. A more relevant case is a spring MVC controller. It is ideal at this level to demonstrate integration testing that includes spring MVC controller as it is mostly used in this work.

Listing 6-6 is used to explain test setup for this testing scenario is part of "Custom App" integration tests. The home page of this system also offers latest new updates and public notices that are interesting at this point. It is expected that when this home page loads, these feeds also load and be well presented on the view. The test is designed to verify that the presented view is a "welcome" view and feeds are loaded as expected.

```

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "classpath:META-INF/applicationContext.test.xml" })
@WebAppConfiguration
@EnableWebMvc
public class DomainServiceWebControllerTests {
    private final String appContextPath = "META-INF/applicationContext.test.xml";
    private TestDataManager testDataManager;
    private DomainDAOImpl domainService;
    @Autowired
    private WebApplicationContext wac;
    private MockMvc mockMvc;
    public void setUp() throws Exception {}
    public void tearDown() throws Exception {}
    @Test
    public void test() throws Exception {
        MvcResult result = mockMvc
            .perform(MockMvcRequestBuilders.get("/"))
            .andExpect(MockMvcResultMatchers.status().isOk())
            .andExpect(MockMvcResultMatchers.view().name("welcome"))
            .andExpect(MockMvcResultMatchers.model().attribute("feeds", testDataManager.findAll(new FeedPost())))
            .andReturn();
    }
}

```

Listing 6-6: Spring MockMvc Test Setup

The first annotation is the common one used to define the test runner. The context configuration annotation is responsible for loading all necessary configuration or beans for use with the tests. Since this is an MVC web application, in order for these tests to run successfully, the context configuration file needs “<mvc:annotation-driven/>” element. In order to test spring MVC controller, spring provides MockMvc package which is a fluent builder testing frameworks with a lot of capabilities around spring web application. MockMvc needs to be configured with the actual component that is being tested or the whole application context where all beans are registered. As can be seen in the listing above, spring auto wiring support is used to inject the web application context which then passed on a test setup to MockMvc service contractor. The test then uses MockMvc’s request builder to prepare a GET request to the root context of the application which is the home page where latest feeds should be loaded. Furthermore, spring’s MockMvc offers a mechanism or matchers that can be used to specify and the expected result.

Shown in Listing 6-7 is another MVC test taken from maturity management system. This test shows more capabilities of the testing framework as used in this research. One of the functionalities that are offered by this system is the ability to register new maternity patients as they begin their maternity journey. This system, however, is not designed to hold general user profile details, but rather an information that is specific to maternity process hence it needs the patient to be registered on the User Account Management System (UAMS). If the patient does not have that account, they can be assisted to create it immediately. Given that the patient does have an account on UAMS, one identification attribute can be used to retrieve that information and complete the registration.

```

@Test
public void when_adding_a_patient_given_uid_should_search_for_user_details_from_uams()
    throws Exception {
    Patient newPatient = null;
    MvcResult result = mockMvc
        .perform(post("/patient/register")
            .param("username", "tester03")
            .param("employmentStatus", "SELF"))
        .andExpect(status().isOk())
        .andExpect(view().name("showpatient"))
        .andExpect(model().attributeExists("patient"))
        .andReturn();

    newPatient = (Patient) result.getModelAndView().getModel().get("patient");
    assertNotNull(newPatient);
    assertEquals(newPatient.getNames(), "Tester03");
    assertEquals(newPatient.getEmploymentStatus(), "SELF");
}

```

Listing 6-7: Spring MockMvc - Patient Registration

The two necessary parameters to create a patient account are passed as request parameters. It is expected that the response status code for that request should be 200 (ok) indicating success which also means that the resource was reached. With the parameters passed to the controller, after its processes it is expected to show the specified MVC view that shows the application details of the patient. The results can always be extracted and be further examined to make sure that the service user test work as intended. The MVC model, in this case, is further interrogated through JUnit assertions to test if the application details are of the patient being registered.

## 6.4 REST API Testing

Some of the key concepts of this research include system integration and interoperability as described in the context of his research. The disparate systems that have been demonstrated and tested so far are used to demonstrate these concepts for the intended audience. It is vital for the purpose of this research to provide a platform where all these disparate systems can be unified and accessed to ease service provisioning. A REST API was therefore developed to offer seamless access to the integrated services. Discussed below is how the API was tested using the REST-Assured testing framework.

```

@org.junit.BeforeClass
public static void BeforeClass() {
    RestAssured.baseURI = "http://192.168.43.12:8181";
    RestAssured.basePath = "/cxf/upservice";
}

```

Listing 6-8: Rest-Assured Test Class Setup

The snippet in Listing 6-8 shows the test setup method that configures REST-Assured with the base URL and path to use in the test fixture. One of the services offered by this REST API is an ability to query user profile from two distributed systems, combine the response and present it to the calling client. This allows the requestor to indicate the filter to retrieve the user profile from any of the systems or both based on the filter.

```
@Test
public void given_valid_username_and_basic_profile_mode_should_return_user_details() {
    String username = "admin";
    response = when()
        .get("/basic/" + username)
        .then().contentType(ContentType.JSON)
        .extract().response();

    ArrayList<Map<String, ?>> jsonAsArrayList = from(response.asString()).get("personalDetails");
    assertNotNull(jsonAsArrayList);
    assertTrue(jsonAsArrayList.get(0).containsKey("firstName"));
    assertTrue(jsonAsArrayList.get(0).containsKey("lastName"));
    assertEquals(jsonAsArrayList.get(0).get("username"), username);
}
```

Listing 6-9 Basic User Profile Test

The test in Listing 6-9 sends a GET request to get a basic profile for “admin” user. The resource that will be invoked by this request is the one that retrieves only basic user information with the document. This means on UAMS system will be invoked by this request. As seen above, the options specified before “then()” option are in this case going to form part of the request while options after “then()” are actually the expectation on the response that will eventually be produced. This test does not specify many options for both requests and expected a response but it extracts the response for further processing.

Since the response is expected to be in JSON format, this means JSONPATH can be used to interrogate this object for further validation. The response string is then passed to JSONPATH’s “from()” method that has capabilities to transform the content into a java map object if it is a JSON object or into a list of map object if the content is a JSON array. In the case above it is expected that the response returns a list of matching records hence a list of map objects is used. It is expected that the list is not null otherwise the test fails. The first map object should have the specified keys and a username key should match the username that was given as a path parameter on the requests. To go further with the test fixture, Listing 6-10 shows another test case.

```

@Test
public void when_getting_a_full_profile_given_a_valid_username__should_return_user_details_and_document() {
    String username = "admin";
    response = when()
        .get("/full/" + username)
        .then()
        .statusCode(200)
        .contentType(ContentType.JSON)
        .extract().response();

    ArrayList<Map<String, ?>> userDetails = from(response.asString()).get("personalDetails");
    ArrayList<Map<String, ?>> documents = from(response.asString()).get("documents");

    assertNotNull(userDetails);
    assertNotNull(documents);
    assertEquals(userDetails.get(0).get("username"), username);
}

```

Listing 6-10: Full User Profile Test

The GET request this time around is sent to a resource that is able to get information from the two systems, combine the responses and return one response. The response string is then manipulated to get two array lists for profile details and documents respectively using JsonPath. JUnit assertions are used to verify that the lists are not null and the username still matches the one that was fed into the request.

## 6.5 Conclusion

This chapter has presented a detailed insight on how each system and its functionalities were tested to ensure that they work as intended. Testing scenarios and levels explained and elaborated by picking relevant functionalities from various systems. Unit testing was performed to ensure that the actual logic implementation of functionalities works as prescribed. Integration testing was then performed to ensure that each system is able to integrate its dependencies and layers and work as whole to offer a full functionality. As system integration was done as the requirement of this work, a REST API that allows access to the integrated services was developed, there was a need to make sure that the API is able to perform all the intended functionalities as expected.

## **Chapter Seven: Result and Analysis**

### **7.1 Introduction**

This chapter presents the overall findings of this research through addressing the research objectives and related research questions. May I reiterate that this research was set to investigate approaches to technically understand and implement integrated interoperable platform that offers integrated services to citizens and other relevant bodies in the context of ICT4D and electronic government. The results were then obtained by following a mixed-methods approach as a methodology that is flexible enough to cater for any kind of requirements in order to absorb key information at any stage of the research. An extensive literature review was conducted to understand the research domain and key principles and technologies in order to obtain the desired results. A quantitative study was also conducted as means to engage with citizens to gather information about their needs as they form part of the problems that this research seeks to overcome. An evolutionary prototyping was employed in developing the prototypes while investigating approaches for technical implementation of ICT systems in the context of this research.

### **7.2 ICT and E-Government in South Africa**

According to literature presented in this work, it is clear that South Africa has been for a long time much aware of the opportunities that ICT holds in providing services to the citizens and other bodies such as private sector and international affairs. South Africa has made major investments in the utilization of ICTs to better deliver services and has established a number of structures to facilitate and administer ICT plans and policies (Matavire and Chigona, 2010; Kaisara and Pather, 2011). Some of the key players in the overall ICT implementation in the country includes Government Information Technology Officers Council (GITOC), State Information Technology Agency, Department of Public Service and Administration (DPSA) and Department of Communication (DoC). A number of other bodies from non-profit organizations (NGOs) and academic institutions have also shown interest in harnessing ICT with developmental plans of the country. This reveals that ICT in South Africa has become a major focus area.

A number of diverse joint projects and initiatives such as those cited in chapter two were established by various stakeholders to investigate and implement necessary fundamentals in order to bring desired solutions. Most of these initiatives aimed at providing broadband access to communities while equipping communal people with ICT tools and necessary skills. As such, there exists a number of centers such as living labs and tele-centers where communal people make use of ICT tools for their personal activities while building and improving their computer literacy skills. One example of such centers is Siyakhula Living Lab (SLL), a joint project between the University of Fort Hare and Rhodes University as described in chapter two. It is worth to mention that the government of South Africa saw a need to establish what is now known as Thusong Service Centers (TSC) as a strategy to deliver information and service to the public citizen in a cost-effective and integrated approach in the context of rural and sustainable development. With the mandate of universal access agency of South Africa, TSCs adopted ICTs to allow digitals access to information and service delivery. Partnership with private companies was then established to elevate the motive of these centers by campaigning for and providing broadband access.

The literature presented in this work also revealed that to date, South Africa has a remarkable number of ICT services publicly available for dissemination of information and service delivery. To mention a few, in the e-service context, the country celebrates the existence of SARS e-filing system, department of Transport's e-Natis systems, department of home affairs e-channel and the e-Disclosure system. This is a testimony that the country has made enormous efforts to better serve its citizen despite the pace and challenges.

It is, however, evident from the literature that even to date there exist challenges in the structures that are set to deal with the communication and interaction between government and citizens. The challenges are said to be part of the contributing factors that hinder success in the anticipated citizen participation in democratic and government processes that are meant for service delivery. HSRC (2016) further indicates that the current mechanisms set to allow citizen participation in government processes are strongly influenced by political dynamics and this makes it even difficult to penetrate communities to disseminate information and deliver services. It is further suggested in one of the 2016 HSRC papers that there is a need to invest and implement proper ICT-based communication and service delivery channels that promote citizen participation and development.

### **7.3 State of Rural Communities**

So far we have provided the findings that relate to South Africa's endeavors to deal with service delivery by making use ICT-based channels. We have also provided an insight of the situations and needs of rural dwellers. Based on the study conducted, it appears that systems that are currently in place for dissemination of information and service delivery in the context of rural development have not reached the desired level of satisfaction. As findings of the study stipulate, there still exist communities where the rate of unemployment is too high and people only rely on urbanized service providers.

Although organizations have made efforts to provide skills to rural inhabitants, this study revealed that there still exist communities where people have very low computer literacy levels which could imply that these people have little or no hands-on experience with ICT-based service. Another very important point to note is that the employment rate in Tyhume region is very low while dwellers constantly have to visit their urban-based service providers. Based on the sampled population for the purpose of this study, about 42% of dwellers go to town on a daily basis while one-third of them go to town at least once a month. This also suggests that dwellers are also confronted the by the transport costs in order to get access to their urbanized service providers.

Understanding the situations in the rural areas afforded this research with scenarios for developing prototypes to demonstrate technical implementation of system integration and interoperability of ICT systems that have potential to accelerate the development of the country. Presented and discussed in the following sections are the results of the technical implementation of this work to address the defined objectives.

### **7.4 Prototype Implementation Results**

#### **7.4.1 System customization**

System Customization in the context of this research as used for dissemination of information is the ability to use one existing system to capture different types of content as needed by various government bodies while reducing costs. This can be well suited in an e-government environment and can reduce costs in the sense that it would eliminate the need for government departments to invest in multiple systems that could offer similar functionalities of information dissemination.



Discussed below are the implantation results for a prototypes that demonstrate system customizability.

One other imported feature to be noted as it applies to all systems is the way in which access control is implemented as to allow the systems to render different pages based on user roles. This statement gives a hint that the system uses a role-based access control approach. Figure 7-1 shows at a basic level the organization of users and LDAP groups that are used as roles as needed by the systems.

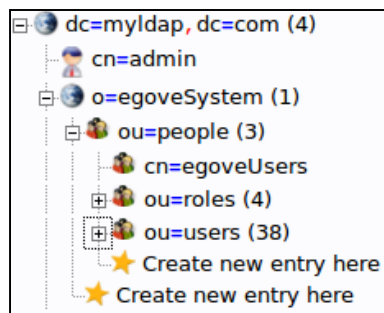


Figure 7-1: LDAP Structure

The two most interesting organizational units (ou) on this figure are roles and users as used during authentication and authorization process. When a user requests to access an authenticated page (URL pattern /web/\*), if they are not logged on yet, they are redirected to a login form in Figure 7-2.

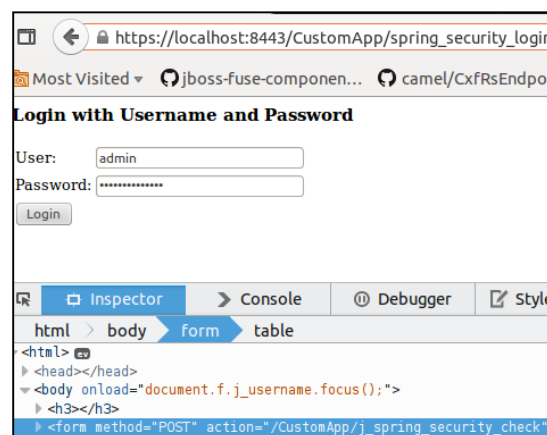


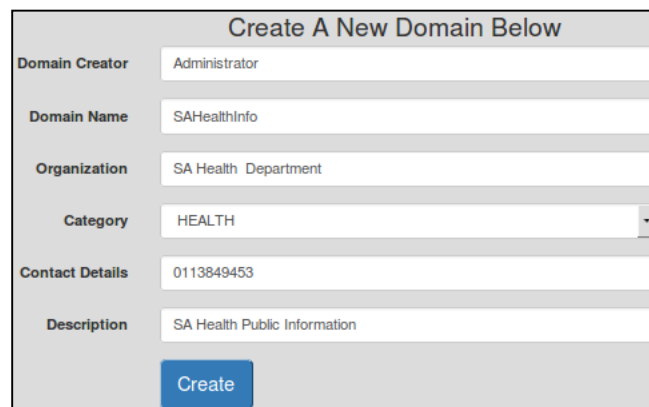
Figure 7-2: Secure Login Page

On submit, the form sends a post request to spring's default authentication/login handler named “\_spring\_security\_check”. As configured on application security contexts, the handler connects

LDAP server to look up the supplied user details from user's organizational unit shown on the LDAP figure 7-2. Having found the matching user details, spring authorities populater will take over and search for roles (groups) to which the user belongs. Based on the results of the authentication handler and the authorities' populater, a relevant informative page will be rendered. while in the context of security, another noticeable point is the use of https. This is so as to ensure a secure environment to access and share information in the context of this research.

As indicated earlier, admin user has an ability to create a domain to group and host entities (organizational or departmental entity records). An example scenario to demonstrate this customization assumes a case where the department of health wants to record all rural clinics in the country and make them publicly available.

To begin with, "SAHealthInfo" domain can be created as indicated in Figure 7-2. The domain name is also used with some naming conversions to create a database for a specific domain which will even be grouped based on the category.



Create A New Domain Below	
Domain Creator	Administrator
Domain Name	SAHealthInfo
Organization	SA Health Department
Category	HEALTH
Contact Details	0113849453
Description	SA Health Public Information
<button>Create</button>	

Figure 7-3: Domain Creation Form

Having successfully created a domain, Figure 7-4 shows how entities within that domain can be created.

The Entity Creation form includes the following fields and table:

Attribute Name	Data Type	Length	Index	NULL?/AI Constraints	Value Options
Id	INT	11	Primary Key	<input checked="" type="checkbox"/> primary key NOT NULL	
AreaName	VARCHAR	255	Unique	<input type="checkbox"/> unique	
Town	VARCHAR	255	None	<input type="checkbox"/>	
Province	VARCHAR	255	None	<input type="checkbox"/>	
ContactDetails	VARCHAR	255	None	<input type="checkbox"/>	
DistanceFromTown	VARCHAR	255	None	<input type="checkbox"/>	

Buttons: Add Row, Remove Row, Create

Figure 7-4: Entity Creation form

This interface gets its motivation from the web UI interface called “PhpMyAdmin” for managing databases and its objects in a flexible manner. The title field in the form is used as a description when displaying the contents of the entity. As in the domain creation process, entity name is also used for naming conversions to create a database table that represents an organizational entity. In cases when an entity is only meant to be used only within the organization and not for public viewing, access type should be set to private. As soon as an entity is created with access type set to “public”, it becomes available for public viewing.

Another major step that administrators or privileged users need to do is to upload entity content on these organizational entities. In order to update content, a relevant service will load all attributes that were specified in the entity creation except for attributes that were marked as auto-incremented values. Figure 7-5 shows a form that is constructed based on the organizational entity’s attributes.

The Content Update Form includes the following fields and button:

Domain	Entity
SAHealthInfo	SARuralClinics
AreaName	<input type="text"/>
Town	<input type="text"/>
Province	<input type="text"/>
ContactDetails	<input type="text"/>
DistanceFromTown	<input type="text"/>
<input type="button" value="Upload"/>	

Figure 7-5: Content Update Form

The “SAHealthInfo” domain was selected to get access to all public entities that belong to it. SARuralClinics entity was then selected and processed by JavaScript to load entity attributes in order to construct a form that captures the needed inputs.

Having many domains under different categories and various entities filled with content within those domains, a user-friendly interface is shown in Figure 7-6 allows users to easily query those organizational entities to get access to public information.

View public Information Below					
<div>HEALTH</div> <div>SAHealthInfo</div> <div>SARuralClinics</div>					
South African Rural Health Centers					
Id	AreaName	Town	Province	ContactDetails	DistanceFromTown
1	Manyengele	Flagstaff	Eastern Cape	0392521198	30km
2	Ntselamanzi Location	Alice	Eastern Cape	060329139	5km
3	Nkozi Location	Flagstaff	Eastern Cape	0392521198	8KM
4	Holly Cross	Lusikisiki	Eastern Cape	0392521198	45KM

Figure 7-6: Public Records Viewing

The user is presented with a drop down list where they select the category from which they want to view information. In this case, “Health” was selected as a category where the desired domain was created. Immediately after selecting a category, a drop-down list with all domains falling under that category is shown on the page. The last drop down list will be shown with all entities that belong to the selected domain. If the user wants to view information with the same domain, there is no need to start from scratch by selecting a category drop-down menu, changing the selected entity will render the relevant records in a table form.

As this kind of system would allow multiple domains with multiple entities to be created, it would provide a seamless easy to use the platform to capture and publicize government records in a reusable manner. This is said to be reusable on the basis that as the system also implements web services, interested parties would only have to create their web service client consume these records and customize them even further for their own use. This would also afford the public with full access to functionalities such as those offered by yellow pages and other private organizations that are focused on dissemination of information.

## 7.4.2 System Integration and Interoperability

System integration and interoperability concepts in this research are implemented to provide a technical demonstration of approaches in which ICT4D and e-government stakeholders in South Africa can develop systems that support secure transactional systems that promote e-government and transparent service delivery. This section presents results of the implemented systems or components for the purpose of this demonstration.

The first requirement is to have a central user account management system that allows public users to create and manage their user profiles. These user profiles should be limited to user details but also there should be a mechanism to allow users to upload some personal documents as they may be required by service providers and even for personal use. To fulfill this requirement, two systems were implemented, one to manage user details and roles while the other one is responsible for managing user documents. When users visit the system, if they already have an account clicking login menu item will show a login form. The sign-up menu item shows the form in Figure 7-7 to capture all necessary details about that user. These details are then saved on the underlying LDAP server that is accessible across all other systems for security and roles.

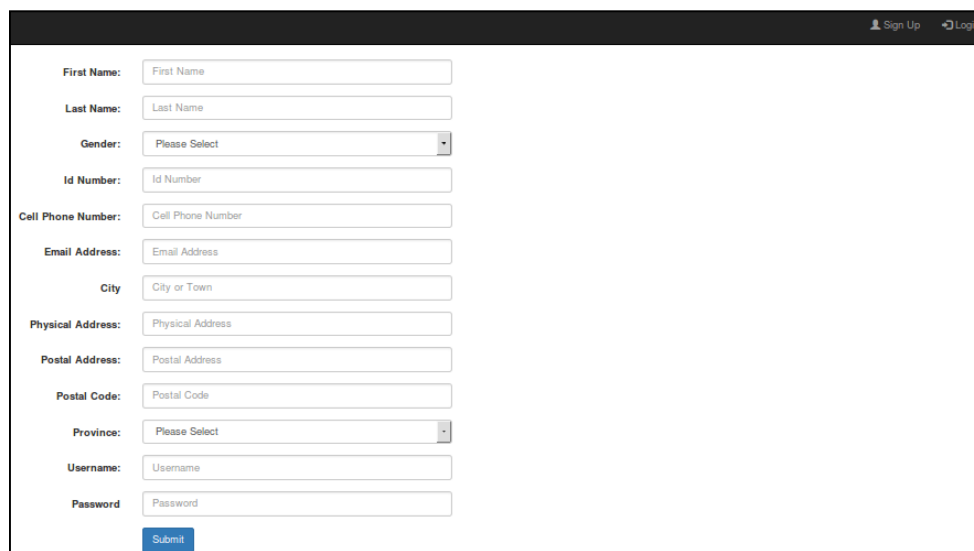
The image shows a web form for account creation. At the top right, there are links for 'Sign Up' and 'Login'. The form contains the following fields: 'First Name' (text input), 'Last Name' (text input), 'Gender' (dropdown menu with 'Please Select' as the current selection), 'Id Number' (text input), 'Cell Phone Number' (text input), 'Email Address' (text input), 'City' (text input), 'Physical Address' (text input), 'Postal Address' (text input), 'Postal Code' (text input), 'Province' (dropdown menu with 'Please Select' as the current selection), 'Username' (text input), and 'Password' (text input). A blue 'Submit' button is located at the bottom left of the form.

Figure 7-7: Account Creation Form

By default, when an account is created using this form, it gets assigned to a “User” group that represents user role. Other roles can later be assigned by someone with relevant privileges. Another important thing to mention is that, these roles are used on this same page to display relevant menu items. For examples, users with “User” role have access to menu items such “My Profile” and “Change password” while users with “Admin” role will have extra menu items such as “Manage Roles” and “Burn Account”.

Another system that is designed to work with the user management systems is a document management system that allows users to management their documents and securely share them with other organizations. Since this application was built using EJBs and JSF, its security implementation is also different and relies on the application server’s security realm. It does,

however, make use of the same LDAP server for user details. Only authenticated users (from LDAP server) can make use of this system as all its URL patterns are secured. For this part, a “testuser” account was created. Figure 7-8 shows an interface to upload documents.

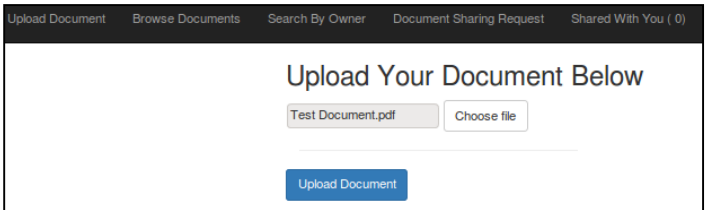


Figure 7-8: Document Upload

Users only need to browse and choose their properly named documents then click the upload button. As HTML input file submits multi-part file content to the server, the content is manipulated to get the necessary details including the document name. The user also has an option to list all of their documents as shown in Figure 7-9.

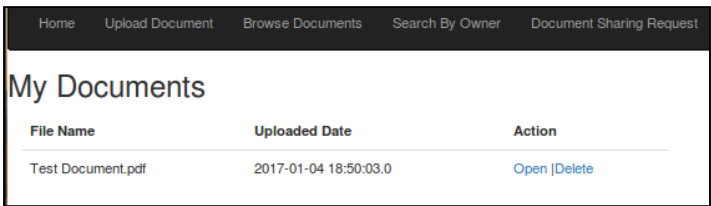


Figure 7-9: Browse Documents

This interface also allows the user to download the document by clicking the “Open” link or delete the document by clicking the “Delete” link on action column. Another interesting feature of this system is the ability for organizations given certain roles to search documents for users and request them. These organizations need to have been explicitly given the role to perform that action and can only browse document names.

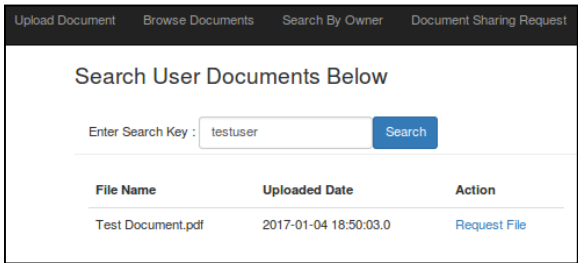
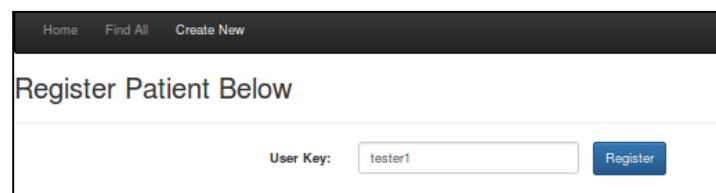


Figure 7-10: Search User Documents

In Figure 7-10, the privileged user is able to search for documents that belong to “testuser”. When the privileged user clicks the “Request File” link, the file owner (“testuser”) will be able to see that request under “Document Sharing Request” menu-item. Only when file owners decide to share the requested file will the requester be able to download the file.

As will be seen later in this chapter, this system can play a vital role in the context of this research as it would reduce costs for traveling to the relevant offices to request such documents and thereafter submit them to other offices. As an example, a case that happens quite often in South Africa is that banks normally need a proof of residence after some time before they can assist the customers. While it is not only banks that need this kind of a document, an individual would have to visit the municipal office to request such documents each time service providers need the document. This would also provide a convenient way to store personal documents while being able to share them with various service providers as long as the document is still valid as certified.

To demonstrate transactional services that seamlessly communicate with other systems including the ones discussed above, a scenario of the department of health maternity monitoring system and department of home affairs was drawn. The maternity record management system requires that patients are registered as soon as they find out about the pregnancy. The registration, however, requires the patient to have an account in the central account management system. Only the identification attributes are used as shown on the figure 7-11 to retrieve the user details from UAMS, perform some processing and store relevant information as needed by the maternity system.



Home Find All Create New

Register Patient Below

User Key:

Figure 7-11: Patient Registration

The search and retrieval of user information by this key is by means of web service communication. This means that the maternity management system has an embedded web service client that communicates with UAMS. If the patient does not have that account, they can be assisted to create it immediately once and for all other future requirements. The next stage after registration would be monthly check-ups that can be added using the form on Figure 7-12.

ID Number	First Name(s)	Last Name	Nurse	Registration Date	Action Now
2517	System	Tester10	Lele Maskiski	2017-01-06 20:33:30.0	<a href="#">Show Records</a> <a href="#">Add Record</a> <a href="#">Register Birth</a>

Place: <input type="text" value="St Methews"/> Nurse: <input type="text" value="Dr Ndebe"/> Infection(s): <input type="text" value="none"/> Weight: <input type="text" value="72"/> Temperature: <input type="text" value="5"/> Medication: <input type="text" value="none"/> Weeks: <input type="text" value="26"/> Delivery: <input type="text" value="NO"/> Comments: <input type="text" value="The patient is in good condition."/>	Delivered: false Health Center: Holly Cross Diagnosis Details: The patient is ok.
---	---

Create

Figure 7-12: Pregnancy Monthly Diagnosis

This is the form that gets shown when a specific patient is selected from a list of patients or when searched by patient's identification attribute. The form captures the diagnosis information for the patient matching the details displayed above the form. The page where this form is displayed also allows the viewing of previous diagnosis information. There are two ways to register birth should a patient reach labor period, it can be registered either by clicking the register birth button at the bottom or by adding check-up records and select "yes" from the drop-down list. This will show another form to capture the details of the labor process including those of the newborn baby.

An interesting feature at this stage is that there are disparate system modules that consume these records and transfer the recent ones with new born babies to the home affairs system for birth certificate application. This happens automatically with no human intervention. This module is able to apply some rules in order to send only eligible applications to the home affairs system.



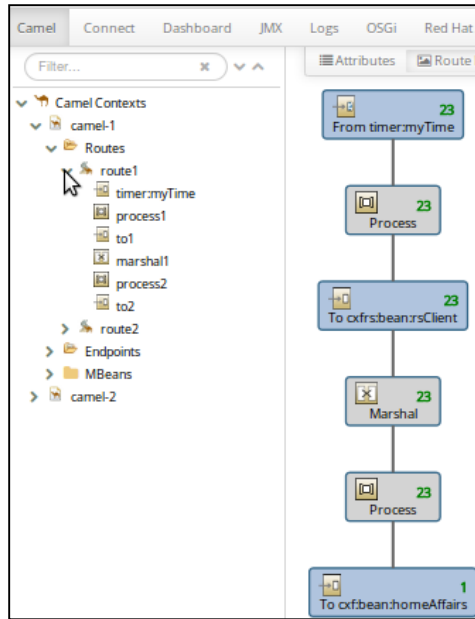


Figure 7-13: Content Polling Manager

Figure 7-13 shows how this polling module works. It is an OSGI bundle that uses a timer routing component to get information from a REST web services and sends it to a SOAP web service. This is by means of apache camel as system integration framework of choice in this research. On Figure 7-13, the module has attempted to retrieve records from maternity system about 23 times, only one matching record was found and submitted. The module is also able to detect records that have already been submitted.

Between the start and the end of this route, there are also other nodes (marshaling and processing) that are implemented to make sure that each service endpoint receives content that it can understand and know which method to invoke. Eventually, the home affairs system will be populated with birth certificate applications for newborn babies as shown in the figure below.

Home Find All		Search Filter Text			
Birth No.	Baby Full Name	Gender	Application Date	Parent Full Name	Action(s)
5	Buzwe Masikisiki	Male	Sat Jan 07 2017 11:58:34 GMT+0200 (SAST)	Tester10 System	Process
7	Test Child	male	Sun Jan 08 2017 09:31:29 GMT+0200 (SAST)	TestUser TestUser	Process

Figure 7-14: List of Applications

Home affairs official will then be able to process applications by selecting one at a time from the list or by searching. To process an application, the official will need full details of the parents and

also possibly some documentation. The process command button on the interface allows the official to view applicant's information as it exists on the two user details systems (UAMS and DMS). Clicking the process button shows the page shown in Figure 7-15.

The screenshot displays a web interface titled "BC Application Details". It is divided into two main sections: "Personal Details" and "Documents".

**Personal Details**

Surname	TestUser
Names	TestUser
Username	testuser
Gender	Male
Cell	0123456789
Email	testuser@gmail.com
Physical Address	Physical Address
PO Box	937
Home Town	Alice
Province	Eastern Cape

**Documents**

Test Document.pdf	<a href="#">Open Document</a>
-------------------	-------------------------------

At the bottom left of the form, there is a blue button labeled "Approve".

Figure 7-15: Parent's/Applicant's Full Profile

Provided that patient uploaded the necessary documents and appear in the document section on the interface, the official can approve the application by clicking the "Approve" button at the bottom of the page.

At this point, all four (4) systems that have been discussed so far are filled with vital information needed to be rightfully accessed by relevant bodies. One other functionality that is not only vital for public users but also key in an e-government platform, is the ability to get all this information from a single point of access in an interactive manner. This does not make all these four systems less important but rather provide a conducive and friendly environment that allow the user to seamlessly consume services offered by that system in a convenient manner. To offer this functionality, OSGI bundles that facilitate routing to and from across all these systems were implemented to expose a REST service. This REST service is then consumed by a Java-backed angular application. Depicted in Figure 7-16 is the navigation menu of that angular application.

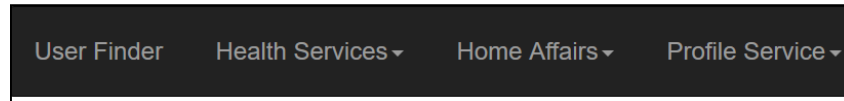


Figure 7-16: Navigation Menu

The navigation is designed such that it groups most of the links that point to the same service domain. This provides a clear categorized navigation that can be easy to use. The profile service drop-down menu item allows users to view and update their records on both user records systems (UAMS and DMS). This also allows users to filter their profile based on what they want at that particular time. It is worth mentioning at this point that, the full profile (including documents) can be retrieved from both underlying systems and presented to the user an integrated manner. Figure 7-17 shows how this integration works.

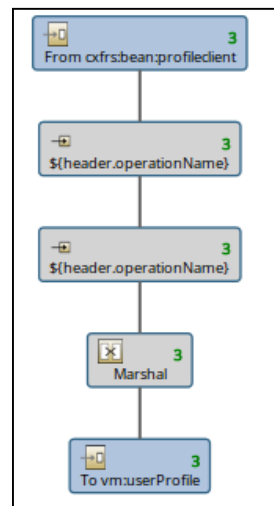


Figure 7-17: User Profile Service Client Route

The OSGI bundle that exposes a rest service as a client of the UAMS service awaits request and use the provided information to properly set headers and data format thereafter forward it to the UAMS service. As explained in the implementation chapter, the node of the route “To vm:userProfile” uses a VM routing component as an indication that the endpoint to which the request should be forwarded is in the context of another OSGI bundle but in the same JVM. This is a key implementation in this work as it promotes separation of concerns, modularity and allows decoupling of service services while they are small manageable pieces.

When the user profile OSGI bundle receives this request it performs conditional routing to determine the endpoint suitable the incoming request as depicted in Figure 7-18.

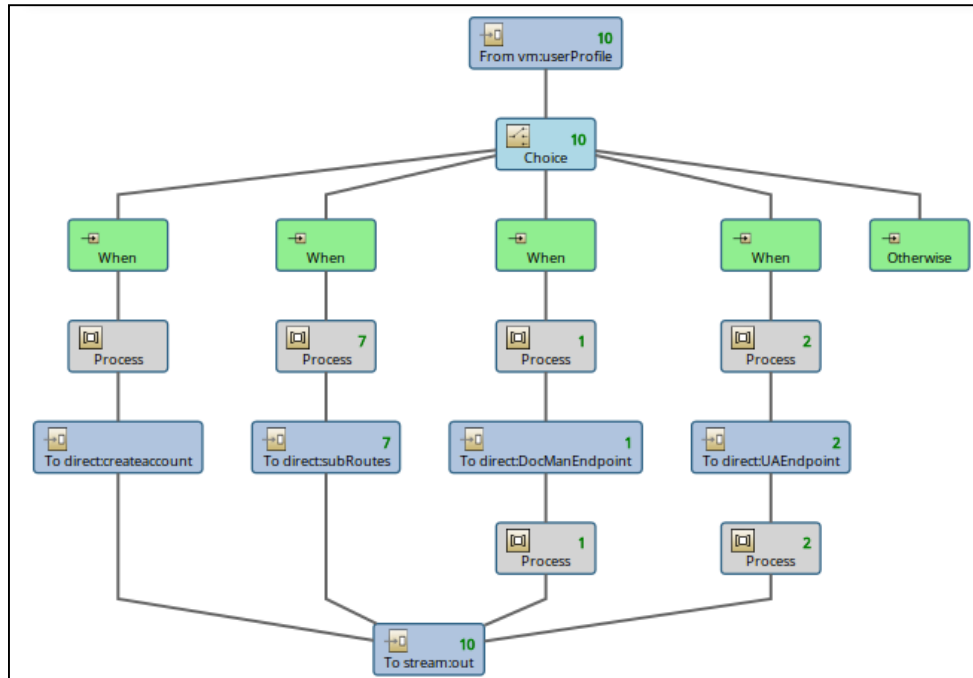


Figure 7-18: User Profile Service Route

There are four types of requests that can be understood by this bundle, a request to create an account, a request to get user document, a request to get only user details and lastly, a request get the full profile which that case it will invoke to endpoints at the same time using multicast component. Any other case will be dealt with on the “otherwise” option or node. The last node “To stream:out” is not really necessary as it only prints out the exchange body on the server side. Whichever node and endpoint that gets called, it is able to produce a response which will be propagated back to the client where it gets accessed by angular application for presentation on the user interface.

This approach also applies on the two bundles for maternity services and home affairs services. The REST services client (API) used for profiles services also provides routing capabilities to both maternity and home affairs systems. This means that this client can send a request to four OSGI bundle context (external bundles) and still be able to get back responses and present it to the user. Shown in Figure 7-19 are two other routes for home affairs and maternity services.

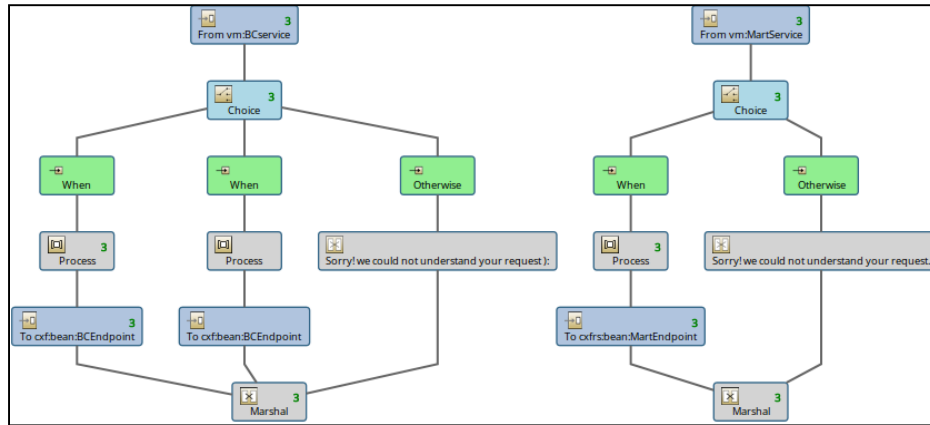


Figure 7-19: BC Service and Maternity Service Routes

The VM component is used again on both routes allow them to accept from requests from different OSGI bundle contexts. They also perform conditional routing using the choice component to determine the relevant service endpoint to which the request be forwarded. On the BCService (birth certificate service for home affairs) route, conditional routing is used to determine the method to invoke in the same endpoint as in any other understandable choices (“when”). All the routes that have been discussed so far, marshaling deals with data format conversions to ensure that responses are in a format that is understandable by the requester or client. It is ideal that these routes are defined in the separate context in order to allow easy maintenance and troubleshooting.

## 7.5 Addressing Objectives and Research Questions

### 7.5.1 Objectives

*The main objective of this research was to develop a prototype that demonstrates how system customization and integration together with interoperability can help in the development of rural development and e-government platforms that enable rural communities to interact with government and/or service providers electronically.*

A customizable system was development and presented in this work as a prototype that implements the cataloging stage of e-government. The customizable feature on this prototype would allow various government bodies to use the same system for disseminating information and cut cost for this level of electronic government while providing a uniform, transparent and convenient platform for citizens to access government’s public records.

The study was conducted as means to understand the situations and needs of rural dwellers. The results of the study were then analyzed, presented and also used to derive scenarios to design and develop other systems and components as means to demonstrate to the defined audience approaches to system integration. A central user account management system (UAMS) was used with department home affairs and department of health as scenarios to demonstrate a seamless interaction between disparate systems. Each of these systems makes use of web services in order to expose entry point or channel for other systems to consume the information. Apache camel was then used to implement the actual system integration following enterprise integration pattern. The final product was a complete integrated platform that allows multiple systems to seamlessly share information during runtime. Having said that, noting the definition of interoperability presented in chapter two as being the ability of two or more disparate systems to exchange and use data, it is therefore safe at this point to say that system interoperability has been addressed in this work. Having successfully explored techniques and principles that allow a successful implementation of system integration and interoperability, and having produced a prototype in this work as a proof of concept, it is again safe to say that solid knowledge and skills have been acquired to address at least one aspect of the Government ICT House of Values; interoperability. Below are the sub-objectives of this work.

- *To understand the current ICT implementation for rural development in South Africa through literature.*

An extensive literature review based on ICT4D, rural development, and the electronic government was conducted to unveil the current state of ICT implementation for development and service provisioning in South Africa. Some of the initiatives found from the literature include projects that a number of stakeholders have embarked on such as rural based-living labs to allow rural citizens to have access ICT tools. The provisioning of computer literacy classes to rural citizens is one the initiatives that were implemented by institutions such as Fort Hare University and Rhodes University to develop and skills rural communities. The government initiative known as Thusong Service Center (TSC) has also been integrated with ICT resources to provide electronic means of service delivery within those centers.

It was also discovered from literature that the government of South Africa adopted ICT for resource allocation, service delivery and full integration with its operations. Various government

departments have since then made a major investment in ICT and managed to produce systems such as SARS e-Filing system and e-Natis among other systems that once earned South Africa recognition in ICT utilization in Africa. However, there still exist challenges in the implementation of a complete and successful electronic government platform in South Africa. Literature presented on this work pointed to the lack of technical skills and the influence politics dynamic to among the contributing factors that hinder the progress and implementation of e-government in South Africa.

- *To investigate the range of needs of rural communities that can be addressed using ICTs.*

A study was conducted to get an understanding of the situations and needs of rural dwellers in South Africa. According to the study, low levels of employment and education are among other challenges that continue to face rural dwellers. The study also revealed that rural communities heavily rely on urbanized service providers for most of their needs. Although there is high unemployment rate, dwellers continue to timeously visit their urbanized service providers which include Banks, health centers, municipality offices and police stations to mention a few.

## **7.6 Conclusion**

In closing, three services (user account/profile management system, maternity management system and birth certificate system) were presented in this chapter with small, independent and manageable OSGI bundles that are able to communicate and share information in real time. This demonstrated a seamless integration of disparate application that represent different organizational or departmental day-to-day functionalities. A single point access to information hosted on multiple systems was created and presented in this chapter as the requirement of this research. This also demonstrated a transparent and automated platform to administer and deliver services. The literature on the use of ICT and electronic government in South Africa has in many occurrences cited a need for the implementation of this kind of integration in most of the South African e-government system. This would then increase citizen participation in transactional and interactive e-government services thereby improving lives of South African citizens.

## **Chapter Eight: Conclusion and Future Work**

### **8.1 Introduction**

The previous chapter has presented the overall findings of this research and has also addressed the research objectives. This chapter concludes and summarizes everything that was covered in this work. The possible future extension will also be presented in this chapter. As this chapter concludes this work, it should be noted that this work was set to investigate and demonstrate approaches to design and develop integrated ICT solutions in order to better service delivery to South Africa citizens, especially those in rural communities. This research gained its motivation from the fact that challenges facing rural dwellers have not been exhaustively addressed by the existing structures that are meant to deliver services and improve the lifestyle of South Africa rural citizens. The purpose of this research was then set to contribute to the development of the country from the ICT4D perspective. The main focus was then put on technical skills to design and develop ICT systems in the context of ICT4D and electronic government.

### **8.2 Conclusion**

This work started off by extensively reviewing the current literature in the rural development, ICT4D, and electronic government space. A number of ICT-based initiatives that South Africa has embarked on as means to improve service delivery and skill citizens were revealed in chapter two. A fair amount of effort was put on underrating e-government concepts and how other countries have adopted and implemented it. Since the main focus of the this work was on the technical skills necessary to design and implement ICT solutions, a number of ICT and Software development concepts and principles were investigated and presented. Security is one of the areas that this research needed to touch basis on because of its importance in this research context although it was not the main focus.

It was through literature that authors came to realization of the fact that, even though South Africa has more than a decade of major investments in the utilization of ICTs for better delivery of services, there is still no fully-fledged e-government platform. To confirm the above findings, electronic government was studies in detail to understand its levels and sectors. It was further



discovered that only two stages or levels of e-government, namely: cataloguing and transactional stage, are evident and have been successfully implemented in South Africa.

For some years the country's e-government implementation has been under scrutiny at least by other scholars, questioning its pace and progress. Nonetheless, South Africa does have number of structures such boards and bodies, standards and frameworks that are set to monitor, guide and regulate ICTs in the public sector, some of these are presented in chapter two. The government ICT-House of values as one of the pieces that South Africa put together as a high-level guide to adhere to when implementing ICTs in the context of e-government was also discovered and studied in this work.

The research went on to conduct a study that aimed at revealing the situations and needs of rural dwellers. Questionnaires were distributed in three areas under Tyhume region of the Eastern Cape province of South Africa. In cases where necessary, interviews were conducted to get and follow up on interesting facts that participants presented. The results were analyzed using R-language and presented in chapter three. Scenarios were drawn to establish use-cases needed to design and implemented the intended prototypes. Eventually, system design was conducted following but not limited to the interaction system design model. The scenarios presented and designed were then implemented. Lastly, the implementation results were analyzed and discussed.

The key terms in this research were system customization as defined here within, system integration and lastly, system interoperability. The main idea behind system customization as per the scope of this research was to demonstrate an approach that allows reusability and sharing of a single system or application amongst different groups or organizations. This would then offer a number of advantages such as reduced effort to maintain multiple applications, reduced budget and provide a uniform platform for information access. For the scope of this research, this kind of customization was then applied only to systems that focus on dissemination of information as the part of the cataloging stage of electronic government. System and integration and interoperability implementation focused more on making various systems and components communicate and share information in an automated manner. With the implementation provided, all three key concepts of this research were successfully demonstrated based on the definition and abstraction provided.

### **8.3 Recommendations**

Based on the literature gathered and presented on this work and the technical exploration of techniques, tools and technologies, we then recommend the following to the target audience which is rural development and e-government stakeholders. In order to drive towards a successful implementation of progressive ICT-based solution to improve service delivery, a call must be made to research institutions, skilled IT specialists, agencies and government officials from various departments to come together and plan, revise standards and frameworks and eventually implement integrated ICT solution as other parts of the world have done so as to witness the fruits of electronic government. These various stakeholders need to work tirelessly not only for specific projects but for the overall e-government platform. A crucial consideration that should be put into this initiative is the continuous involvement of citizens in order to plan, design and implement solutions that are relevant to the needs of the citizens. The public sector needs to establish a plan that will ensure that IT practitioners timeously improve their technical skills and knowledge. Each ICT product produced at any milestone must follow standards and principle that allow seamless integration with existing and future solutions.

The government ICT house of values as outlined and adopted by the public sector comprises very broad areas that are crucial in the progression of an electronic government platform. It could be ideal for the public sector to break these down and seek a continuous support accompanied by research and implementation that caters for the evolving requirements and technology. As an example, using this work as a proof of concept, investigation of system integration and interoperability from a technical point of view was successfully conducted in this work and yielded positive results for a start. It is this kind of work in which the public sector needs to continuously invest as means deal with the interoperability part of the Government ICT House of Values. Having a similar idea for the rest of other parts of Government ICT House of Values could lend the public sector knowledge and technical skills to move forward with the e-government implementation.

### **8.4 Future Work**

This research produced a prototype system that allows multiple systems to communicate and share information. As presented in the results chapter, one entry point to consume information from

multiple systems was then defined with a usable web interface. This was made possible by defining a REST service that listens for a client request and dispatches them to relevant endpoints then combines the responses and presents them to the client. Future extensions around this area may be to develop a mobile application that will consume this REST service.

## **8.5 Conclusion**

This chapter has given the overall summary of this work followed recommendations that are based on the findings of the literature review presented herewith and the technical exploration and demonstration of tools and techniques to design and implement integrated ICT systems. Possible future extensions include the development of a mobile application that can act as a client to REST API that acts as an entry point to communicate and consume information in an integrated manner.

## References

- Adams, A. and Cox, A. (2008) 'Questionnaires, in-depth interviews and focus groups'. Available at: <http://oro.open.ac.uk/11909/> (Accessed: 16 January 2017).
- Ahmad, A., Maynard, S. and Park, S. (2014) 'Information security strategies: towards an organizational multi-strategy perspective', *Journal of Intelligent Manufacturing*. Available at: <http://link.springer.com/article/10.1007/s10845-012-0683-0> (Accessed: 9 September 2015).
- Alonso, G. *et al.* (2004) *Web Services*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-662-10876-5.
- Andersen, K. V. and Henriksen, H. Z. (2006) 'E-government maturity models: Extension of the Layne and Lee model', *Government Information Quarterly*, 23(2), pp. 236–248. doi: 10.1016/j.giq.2005.11.008.
- Andress, J. (2014) *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice*. Elsevier Science. Available at: <https://books.google.com/books?hl=en&lr=&id=9NI0AwAAQBAJ&pgis=1> (Accessed: 9 September 2015).
- Backus, M. (2001) 'E-governance in Developing Countries', *IICD Research Brief*. Available at: <http://editor.iicd.org/files/report3.doc> (Accessed: 18 January 2015).
- Bansler, J. P. and Havn, E. (2010) 'Pilot implementation of health information systems: Issues and challenges', *International Journal of Medical Informatics*, 79(9), pp. 637–648. doi: 10.1016/j.ijmedinf.2010.05.004.
- Basu, S. (2004) 'E-government and developing countries: an overview', *International Review of Law, Computers & Technology*, 18(January 2015), pp. 109–132. doi: 10.1080/13600860410001674779.
- Benatallah, B. and Nezhad, H. (2008) 'Service oriented architecture: Overview and directions', *Advances in Software Engineering*. Available at: [http://link.springer.com/chapter/10.1007/978-3-540-89762-0\\_4](http://link.springer.com/chapter/10.1007/978-3-540-89762-0_4) (Accessed: 1 May 2015).
- Bertossi, L. and Bravo, L. (2005) 'Consistent query answers in virtual data integration systems', *Inconsistency tolerance*. Available at: [http://link.springer.com/chapter/10.1007/978-3-540-30597-2\\_3](http://link.springer.com/chapter/10.1007/978-3-540-30597-2_3) (Accessed: 5 June 2015).
- Brazhnik, O. and Jones, J. F. (2007) 'Anatomy of data integration.', *Journal of biomedical informatics*, 40(3), pp. 252–69. doi: 10.1016/j.jbi.2006.09.001.
- Burch, S. (2007) 'Knowledge sharing for rural development: challenges, experiences and methods'. Available at: <http://agris.fao.org/agris-search/search/display.do?f=2013/GB/GB2013203400034.xml;GB2013203404> (Accessed: 11 April 2014).
- Burton, S. (2011) 'Thusong Service Centres Towards a Broader Planning Perspective', pp. 1–9.
- Calì, A. and Calvanese, D. (2013) 'Data integration under integrity constraints', *Seminal Contributions to ...*. Available at: <http://link.springer.com/chapter/10.1007/978-3-642-36926->

1\_27 (Accessed: 5 June 2015).

Che, M. and Tuo, M. (2016) 'Server Program Analysis Based on HTTP Protocol', *MATEC Web of Conferences*. Available at: [http://www.matec-conferences.org/articles/mateconf/abs/2016/26/mateconf\\_mmme2016\\_05023/mateconf\\_mmme2016\\_05023.html](http://www.matec-conferences.org/articles/mateconf/abs/2016/26/mateconf_mmme2016_05023/mateconf_mmme2016_05023.html) (Accessed: 25 January 2017).

Chen, D. (2013) 'Enterprise Interoperability', *Iwei*, 144. doi: 10.1007/978-3-642-36796-0.

Chetty, M., Tucker, W. and Blake, E. (2003) 'Using voice over IP to bridge the digital divide: a critical action research approach'. Available at: <http://repository.uwc.ac.za/xmlui/handle/10566/479> (Accessed: 24 June 2015).

Cloete, F. (2012) 'Sabinet - E-government lessons from South Africa 2001-2011 : institutions, state of progress and measurement : Section II : Country perspectives on e-government emergence', *SA ePublications*, pp. 128–142. Available at: [http://reference.sabinet.co.za/sa\\_epublication\\_article/afjic\\_n12\\_a8](http://reference.sabinet.co.za/sa_epublication_article/afjic_n12_a8) (Accessed: 5 August 2015).

Coetzee, H., Toit, I. Du and Herselman, M. (2012) 'Living Labs in South Africa: An analysis based on five case studies'. Available at: <http://researchspace.csir.co.za/dspace/handle/10204/6082> (Accessed: 31 August 2014).

Conradie, D. P. *et al.* (2010) 'Communicatio : South African Journal for Communication Theory Using information and communication technologies ( ICTs ) for deep rural development in South Africa', (January 2015), pp. 37–41. doi: 10.1080/02500160308538027.

Conradie, D. P., Morris, C. and Jacobs, S. J. (2003) 'Using information and communication technologies (ICTs) for deep rural development in South Africa', *Communicatio*, 29(1–2), pp. 199–217. doi: 10.1080/02500160308538027.

Cruz, I. and Xiao, H. (2005) 'The role of ontologies in data integration', ... *intelligent systems for electrical engineering and ....* Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.4933&rep=rep1&type=pdf> (Accessed: 7 June 2015).

Curry, E. (2005) *Middleware for Communications*. John Wiley & Sons. Available at: [http://books.google.com/books?hl=en&lr=&id=bh8\\_0FD3qgQC&pgis=1](http://books.google.com/books?hl=en&lr=&id=bh8_0FD3qgQC&pgis=1) (Accessed: 5 May 2015).

Curry, E. (2012) 'System of systems information interoperability using a linked dataspace', in *Proceedings - 2012 7th International Conference on System of Systems Engineering, SoSE 2012*, pp. 101–106. doi: 10.1109/SYSoSE.2012.6384200.

Dalvit, L., Siebörger, I. and Thinyane, H. (2012) 'The expansion of the Siyakhula Living Lab: a holistic perspective', *e-Infrastructure and e-Services for ....* Available at: [http://link.springer.com/chapter/10.1007/978-3-642-29093-0\\_22](http://link.springer.com/chapter/10.1007/978-3-642-29093-0_22) (Accessed: 22 June 2015).

Danish, D. (2006) 'The Failure of E-Government in Developing Countries: A Literature Review', *The Electronic Journal of Information Systems in Developing Countries*, 26, pp. 1–10.

Dayanidhi, M. (2005) *E-government Toolkit for Developing Countries*.

- Doan, A., Halevy, A. and Ives, Z. (2012) *Principles of Data Integration*. Elsevier. Available at: <https://books.google.com/books?hl=en&lr=&id=s2YCKGrO10YC&pgis=1> (Accessed: 3 June 2015).
- Dong, X., Halevy, A. and Yu, C. (2007) 'Data integration with uncertainty', ... *international conference on Very large data* .... Available at: <http://dl.acm.org/citation.cfm?id=1325930> (Accessed: 3 June 2015).
- Emmersberger, C. and Springer, F. (2013) 'Tutorial: Open source enterprise application integration-introducing the event processing capabilities of apache camel', *Proceedings of the 7th ACM international* .... Available at: <http://dl.acm.org/citation.cfm?id=2488269> (Accessed: 20 May 2016).
- Farooq, M. U. *et al.* (2015) 'A Critical Analysis on the Security Concerns of Internet of Things (IoT)', *International Journal of Computer Applications*, 111(7), pp. 1–6. Available at: [http://www.researchgate.net/publication/272488652\\_A\\_Critical\\_Analysis\\_on\\_the\\_Security\\_Concerns\\_of\\_Internet\\_of\\_Things\\_\(IoT\)](http://www.researchgate.net/publication/272488652_A_Critical_Analysis_on_the_Security_Concerns_of_Internet_of_Things_(IoT)) (Accessed: 12 September 2015).
- Fong, E. (2007) *e-Government Interoperability: Overview, UNDP Regional Centre in Bangkok With the support of: IBM Oracle*.
- Frantz, R. (2015) 'A methodology to evaluate the maintainability of enterprise application integration frameworks', *International Journal of* .... Available at: <http://www.inderscienceonline.com/doi/abs/10.1504/IJWET.2015.073951> (Accessed: 24 May 2016).
- Frantz, R. and Corchuelo, R. (2012) 'A software development kit to implement integration solutions', *Proceedings of the 27th Annual ACM* .... Available at: <http://dl.acm.org/citation.cfm?id=2232042> (Accessed: 24 May 2016).
- Gant, J. P. (2008) 'Electronic Government for Developing Countries', *International Telecommunications Union*, (August), p. 59.
- Gardiner, M. (2008) 'Education in rural areas', *Issues in education policy*. Available at: [http://www.cepd.org.za/files/pictures/Education in Rural Areas 2008.pdf](http://www.cepd.org.za/files/pictures/Education%20in%20Rural%20Areas%202008.pdf) (Accessed: 17 May 2014).
- Gatautis, R., Kulvietis, G. and Vitkauskaitė, E. (2009) 'Lithuanian eGovernment Interoperability Model', *Inžinerinė ekonomika Engineering economics*, 62(2), pp. 38–48. Available at: <http://internet.ktu.lt/lt/mokslas/zurnalai/inzeko/62/1392-2758-2009-2-62-38.pdf>.
- GCIS (2017) *About Thusong Service Centres: Services, objectives and mandate*. Available at: <http://www.thusong.gov.za/about/what/index.htm> (Accessed: 13 January 2017).
- Goncalves, A. (2013) *Beginning Java EE 7*. Berkeley, CA: Apress. doi: 10.1007/978-1-4302-4627-5.
- Goswami, S. (2014) 'ICT: Sustainable Development', *JOURNAL OF INDIAN MANAGEMENT*. Available at: [http://www.scms.edu.in/JIM/pdf/2014/SCMS Journal Jan-Mar 2014.pdf#page=127](http://www.scms.edu.in/JIM/pdf/2014/SCMS%20Journal%20Jan-Mar%202014.pdf#page=127) (Accessed: 11 April 2014).
- Graham, M. (2010) 'ICT and Development'. Available at:

<http://ictlogy.net/bibliography/reports/projects.php?idp=1597> (Accessed: 11 April 2014).

Guijarro, L. (2007) 'Interoperability frameworks and enterprise architectures in e-government initiatives in Europe and the United States', *Government Information Quarterly*, 24(1), pp. 89–101. doi: 10.1016/j.giq.2006.05.003.

Gumbo, S. and Thinyane, H. (2012) 'Living lab methodology as an approach to innovation in ICT4D: The Siyakhula Living Lab experience', *Proceedings of the IST-Africa 2012 Conference*, 24. Available at:

[http://www.siyakhulall.com/sites/default/files/ISTAfrica\\_Paper\\_ref\\_18\\_doc\\_4809\\_0.pdf](http://www.siyakhulall.com/sites/default/files/ISTAfrica_Paper_ref_18_doc_4809_0.pdf) (Accessed: 31 August 2014).

Hafez, M., Ali, H. and Hegazy, O. (2013) 'A Statistical Data Fusion Technique in Virtual Data Integration Environment', *International Journal of Data Mining and ...*. Available at: <http://www.airccse.org/journal/ijdkp/papers/3513ijdkp03.pdf> (Accessed: 5 June 2015).

He, W. and Xu (2012) 'Integration of Distributed Enterprise Applications: A Survey', *IEEE Transactions on Industrial Informatics*, 10(1), pp. 35–42. doi: 10.1109/TII.2012.2189221.

He, W. and Xu, L. Da (2014) 'Integration of distributed enterprise applications: a survey', *Industrial Informatics, IEEE Transactions on*. Available at: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=6165353](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6165353) (Accessed: 22 May 2016).

Hu, V., Ferraiolo, D. and Kuhn, D. (2006) *Assessment of access control systems*. Available at: <http://www.fmpik.gov.ba/admin/files/2013604590.pdf> (Accessed: 6 May 2015).

Husák, M. *et al.* (2015) 'Network-based HTTPS Client Identification Using SSL/TLS Fingerprinting'.

Huysman, M. H. and Wit, D. de (2002) *Knowledge Sharing in Practice*. Springer. Available at: <http://books.google.com/books?hl=en&lr=&id=Ta1f80EGVSgC&pgis=1> (Accessed: 11 April 2014).

Ikeda, M., Ashley, K. D. and Chan, T.-W. (eds) (2006) *Intelligent Tutoring Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science). doi: 10.1007/11774303.

ISO/IEC and IEEE (2010) 'ISO/IEC/IEEE 24765:2010 - Systems and software engineering -- Vocabulary', *ISO/IEC IEEE*, 2010, p. 410. doi: 10.1109/IEEESTD.2010.5733835.

Jeelani, Z. (2013) 'AN INSIGHT OF SSL SECURITY ATTACKS', *International Journal of Research in Engineering & Applied Sciences*, 3(52). Available at: <http://www.euroasiapub.org> (Accessed: 24 January 2017).

Johner, H. *et al.* (1998) 'Understanding LDAP', *Contract*, p. 177.

Joshi, J. and Aref, W. (2001) 'Security models for web-based applications', *Communications of the ...*. Available at: <http://dl.acm.org/citation.cfm?id=359224> (Accessed: 13 April 2015).

Jr., W. J. M. and Elmagarmid, A. K. (2002) *Advances in Digital Government: Technology, Human Factors, and Policy*. Springer Science & Business Media. Available at: <https://books.google.com/books?hl=en&lr=&id=mKMrZyDSZkC&pgis=1> (Accessed: 4 June 2015).

2015).

Kaisara, G. and Pather, S. (2011) 'The e-Government evaluation challenge: A South African< i>Batho Pele</i>-aligned service quality approach', *Government Information Quarterly*. Available at: <http://www.sciencedirect.com/science/article/pii/S0740624X10001401> (Accessed: 18 January 2015).

Kalin, M. (2009) *Java web services: up and running*. 'O'Reilly Media, Inc.' doi: 10.1093/hmg/ddp095.

Kamlesh, V. and Ahmad, S. (2008) 'Evaluating Evolutionary Prototyping for Customizable Generic Products in Industry (TAT AB)'. Available at: <http://www.diva-portal.org/smash/get/diva2:829677/FULLTEXT01.pdf> (Accessed: 3 April 2018).

Ke, W. and Wei, K. K. (2004) 'Successful e-government in Singapore', *Communications of the ACM*. ACM, 47(6), pp. 95–99. doi: 10.1145/990680.990687.

Khane, C. and Siebörger, I. (2011) 'The Siyakhula living lab: a holistic approach to rural development through ICT in rural South Africa', *Steyn, J., Van Belle, J. ....* Available at: <http://www.igi-global.com/chapter/content/66139> (Accessed: 17 May 2014).

Kleine, D. and Unwin, T. (2009) 'Technological Revolution, Evolution and New Dependencies: what's new about ict4d?', *Third World Quarterly*, 30(5), pp. 1045–1067. doi: 10.1080/01436590902959339.

Kolb, P. (2008) 'Realization of eai patterns with apache camel'. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.214.3580&rep=rep1&type=pdf> (Accessed: 22 May 2016).

Kosanke, K., Vernadat, F. and Zelm, M. (2014) 'Means to enable Enterprise Interoperation : CIMOSA Object Capability Profiles and CIMOSA Collaboration View', pp. 3292–3299.

Krauss, K. (2012) 'Towards self-emancipation in ICT for development research: narratives about respect, traditional leadership and building networks of friendships in rural South Africa', *The African Journal of information ....* Available at: [http://digitalcommons.kennesaw.edu/ajis/vol4/iss2/1/?utm\\_source=digitalcommons.kennesaw.edu%2Fajis%2Fvol4%2Fiss2%2F1&utm\\_medium=PDF&utm\\_campaign=PDFCoverPages](http://digitalcommons.kennesaw.edu/ajis/vol4/iss2/1/?utm_source=digitalcommons.kennesaw.edu%2Fajis%2Fvol4%2Fiss2%2F1&utm_medium=PDF&utm_campaign=PDFCoverPages) (Accessed: 24 June 2015).

Langegger, A., Wöß, W. and Blöchl, M. (2008) *A semantic web middleware for virtual data integration on the web*. Available at: [http://link.springer.com/chapter/10.1007/978-3-540-68234-9\\_37](http://link.springer.com/chapter/10.1007/978-3-540-68234-9_37) (Accessed: 5 June 2015).

Layne, K. and Lee, J. (2001) 'Developing fully functional E-government: A four stage model', *Government Information Quarterly*, 18, pp. 122–136. doi: 10.1016/S0740-624X(01)00066-1.

Lee, S., Tan, X. and Trimi, S. (2005) 'Current practices of leading e-government countries', *Communications of the ACM*. Available at: <http://dl.acm.org/citation.cfm?id=1089112> (Accessed: 14 April 2014).

Li, F. *et al.* (2012) 'Research status and development trends of access control model'. Available at: [http://en.cnki.com.cn/Article\\_en/CJFDTOTAL-DZXU201204029.htm](http://en.cnki.com.cn/Article_en/CJFDTOTAL-DZXU201204029.htm) (Accessed: 5 May



2015).

Li, Z., Bi, J. and Wang, S. (2013) 'HTTP-CCN gateway: Adapting HTTP protocol to Content Centric Network', *Network Protocols (ICNP)*, 2013 21st IEEE. Available at: <http://ieeexplore.ieee.org/abstract/document/6733636/> (Accessed: 25 January 2017).

Loukis, E. N. and Charalabidis, Y. K. (2013) 'An empirical investigation of information systems interoperability business value in European firms', *Computers in Industry*. Elsevier B.V., 64(4), pp. 412–420. doi: 10.1016/j.compind.2013.01.005.

Madoukh, S. and Baraka, R. (2013) 'A SOA-Based e-Government Data Integration', *iajet.org*. Available at: [http://www.iajet.org/iajet\\_files/vol.3/no.3/watermark/2.pdf](http://www.iajet.org/iajet_files/vol.3/no.3/watermark/2.pdf) (Accessed: 16 April 2014).

Matavire, R. and Chigona, W. (2010) 'Challenges of eGovernment project implementation in a South African context', *The Electronic Journal* .... Available at: <http://www.ejise.com/issue/download.html?idArticle=673> (Accessed: 18 January 2015).

Maumbe, B. M., Owei, V. and Alexander, H. (2008) 'Questioning the pace and pathway of e-government development in Africa: A case study of South Africa's Cape Gateway project', *Government Information* .... Available at: <http://www.sciencedirect.com/science/article/pii/S0740624X08000440> (Accessed: 18 January 2015).

McIver, W. J. and Elmagarmid, A. K. (eds) (2002) *Advances in Digital Government*. Boston, MA: Springer US (Advances in Database Systems). doi: 10.1007/b116295.

Medjahed, B. *et al.* (2003) 'Business-to-business interactions: Issues and enabling technologies', *VLDB Journal*, 12(1), pp. 59–85. doi: 10.1007/s00778-003-0087-z.

Moleki, G. F. (2007) *MIOsv4.12007*. Available at: <http://www.sita.co.za/standard/MIOsv4.12007.pdf> (Accessed: 18 May 2014).

Moseley, M. J. (2003) *Rural development : principles and practice*. London [u.a.]: SAGE.

Mphidi, H. (2008) 'Digital divide and e-governance in South Africa', *Research, Innovation and Partnerships*. Available at: [http://www.ais.up.ac.za/digi/docs/mphidi\\_paper.pdf](http://www.ais.up.ac.za/digi/docs/mphidi_paper.pdf) (Accessed: 21 June 2015).

zur Muehlen, M., Nickerson, J. V. and Swenson, K. D. (2005) 'Developing web services choreography standards—the case of REST vs. SOAP', *Decision Support Systems*, 40(1), pp. 9–29. doi: 10.1016/j.dss.2004.04.008.

Ngomane, T. (2012) 'Rural Development in South Africa: The role of Agriculture'.

Nkwinti, G. E. (2011) *STRATEGIC PLAN 2011-2014*. Available at: <http://www.ruraldevelopment.gov.za/phocadownload/Strategic-Plan/RDLR-STRAT-PLAN2011.pdf> (Accessed: 17 May 2014).

Olden, E. M. (2002) 'Security and access management system for web-enabled and non-web-enabled applications and content on a computer network'. Available at: <http://www.google.com/patents/US6460141> (Accessed: 13 April 2015).

Pascal, L. (ed.) (2014) *Advances in Communication Technology and Application*. WIT Press. Available at: <http://books.google.com/books?hl=en&lr=&id=R94kBQAAQBAJ&pgis=1> (Accessed: 30 April 2015).

Patil, N., Kshirsagar, M. and Jaypal, P. (2014) 'Enterprise Application Integration using Service Oriented Architecture with Generic Pattern'. Available at: <http://inpressco.com/wp-content/uploads/2014/10/Paper813540-3545.pdf> (Accessed: 30 April 2015).

Pautasso, C., Zimmermann, O. and Leymann, F. (2008) 'Restful web services vs. big web services: making the right architectural decision', *Proceedings of the 17th ...*. Available at: <http://dl.acm.org/citation.cfm?id=1367606> (Accessed: 19 June 2015).

Pierce, J. (2014) *Integration Frameworks and Enterprise Integration Patterns*. Available at: <https://www.credera.com/blog/technology-insights/java/integration-frameworks-enterprise-integration-patterns/> (Accessed: 22 April 2016).

Ponniah, P. (2011) *Data Warehousing Fundamentals for IT Professionals*. John Wiley & Sons. Available at: [https://books.google.com/books?hl=en&lr=&id=aIDjcF9r\\_hgC&pgis=1](https://books.google.com/books?hl=en&lr=&id=aIDjcF9r_hgC&pgis=1) (Accessed: 5 June 2015).

Puustjärvi, J. (2004) 'Using One-Stop Portal in Integrating e-Learning Systems', *Advanced Technology for Learning*. ACTA Press, 1(2). doi: 10.2316/Journal.208.2004.2.208-0803.

Rafique, S. and Humayun, M. (2015) 'Web application security vulnerabilities detection approaches: A systematic mapping study', ... (*SNPD*), 2015 16th ... Available at: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=7176244](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7176244) (Accessed: 10 September 2015).

Rogerson Cdmr, C. M. (2014) 'Bulletin of GeoGraphy. Socio-economic Series Bulletin of GeoGraphy. Socio-economic Series reframing place-based economic development in South africa: the example of local economic development', *Bulletin of Geography. Socio-economic Series*, (24), pp. 203–218. doi: 10.2478/bog-2014-0023.

Ruxwana, N. (2010) 'ICT applications as e-health solutions in rural healthcare in the Eastern Cape Province of South Africa', ... *management journal*. Available at: <http://search.informit.com.au/documentSummary;dn=961281654549031;res=IELAPA> (Accessed: 24 June 2015).

Scott, M. (2010) *Investigation and development of an e judiciary service for a citizen oriented judiciary system for rural communities*. University of Fort Hare. Available at: <http://ufh.netd.ac.za/handle/10353/275> (Accessed: 31 August 2014).

Scott, M., Thinyane, M. and Terzoli, A. (2009) 'Implementation of an e-Judiciary Service for Traditional Justice Administration in Dwesa-Cwebe', *SATNAC 2009*. Available at: <http://www.satnac.org.za/proceedings/2009/papers/software/Paper 51.pdf> (Accessed: 31 August 2014).

Segole, J. (2010) *Government-Wide Enterprise Architecture (GWEA) Framework Implementation Guide*.

Seifert, J. (2003) 'A primer on e-government: Sectors, stages, opportunities, and challenges of online governance'. Available at: <http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA445492>

(Accessed: 21 January 2015).

Sekyere, E., Tshitiza, O. and Hart, T. (2016) 'Levering m-governance innovations for active citizenship engagement'. Available at: <http://repository.hsra.ac.za/handle/20.500.11910/10214> (Accessed: 7 December 2016).

SITA (2014) *Government's IT House of Values*. Available at: [http://www.sita.co.za/about/abt\\_housevalues.htm](http://www.sita.co.za/about/abt_housevalues.htm) (Accessed: 18 May 2014).

Suda, B. (2003) 'SOAP Web Services', *Retrieved June*. Available at: <http://suda.co.uk/publications/MSc/brian.suda.thesis.pdf> (Accessed: 19 June 2015).

Swift, J. (2009) 'Notes on Traditional Knowledge, Modern Knowledge and Rural Development', *The IDS Bulletin*, 10(2), pp. 41–43. doi: 10.1111/j.1759-5436.1979.mp10002007.x.

Thankachan, A., Ramakrishnan, R. and Kalaiarasi, M. (2014) 'A survey and vital analysis of various state of the art solutions for web application security', in *International Conference on Information Communication and Embedded Systems (ICICES2014)*. IEEE, pp. 1–9. doi: 10.1109/ICICES.2014.7033786.

Thomas, I. *et al.* (2006) '(12) United States Patent', 2(12).

TOGAF (2014) *TOGAF® / The Open Group South Africa*. Available at: <http://opengroup.co.za/togaf> (Accessed: 17 April 2014).

Torun, A. (2002) 'Using Schema and Data Integration Technique to Integrate Spatial and Non-Spatial Data: Developing Populated Places DB of Turkey (PPDB\_T)', *INTERNATIONAL ARCHIVES OF ...*. Available at: <http://www.hgk.msb.gov.tr/images/egitim/8a535c0282a55c4.pdf> (Accessed: 5 June 2015).

Wanjiku, R. (2008) *Kenyan Judicial Sector Aims to Improve IT / CIO*. Available at: <http://www.cio.com/article/2436419/it-organization/kenyan-judicial-sector-aims-to-improve-it.html> (Accessed: 31 August 2014).

Wauters, P. (2006) 'Benchmarking e-government policy within the e-Europe programme', *Aslib Proceedings*. Edited by B. Gunter. Emerald Group Publishing Limited, 58(5), pp. 389–403. doi: 10.1108/00012530610692348.

Wyllie, D. and Davies, J. (2015) 'Role of data warehousing in healthcare epidemiology', *Journal of Hospital Infection*. Available at: <http://www.sciencedirect.com/science/article/pii/S0195670115000523> (Accessed: 5 June 2015).

Xiao, H., Cruz, I. and Hsu, F. (2004) 'Semantic Mappings for the Integration of XML and RDF Sources', *Proc. of IIWEB-2004*. Available at: <http://www.cs.uic.edu/~advis/publications/dataint/iiweb04.pdf> (Accessed: 7 June 2015).

Zhang, D. and Nunamaker, J. F. (2003) 'Powering E-Learning In the New Millennium: An Overview of E-Learning and Enabling Technology', *Information Systems Frontiers*. Kluwer Academic Publishers, 5(2), pp. 207–218. doi: 10.1023/A:1022609809036.

Zhang, X. *et al.* (2010) 'Applying evolutionary prototyping model for eliciting system requirement of meat traceability at agribusiness level', *Food Control*, 21(11), pp. 1556–1562.

doi: 10.1016/j.foodcont.2010.03.020.

## Appendix

### A. Document management System - FTP Server connection

```
import java.io.IOException;
import org.apache.commons.net.ftp.FTPClient;

public class FTPConnection {
    private String ftpAddress = "localhost";
    private int port = 21;
    private String user = "admin";
    private String pass = "lizo@200903375";

    public FTPClient connect() {
        FTPClient ftpClient = new FTPClient();
        try {
            ftpClient.connect(ftpAddress, port);
            ftpClient.login(user, pass);
            ftpClient.enterLocalPassiveMode();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return ftpClient;
    }
}
```

## B. Document management System – File Upload

```
public boolean addFile(FileInfo file, String fileSource, InputStream istream) {
    FTPClient ftpClient = new FTPConnection().connect();
    InputStream inputstream = istream;
    boolean done = false;
    if (ftpClient.isConnected()) {
        if (inputstream != null) {
            try {
                ftpClient.setFileType(FTP.BINARY_FILE_TYPE);
                ftpClient.setFileTransferMode(FTP.BINARY_FILE_TYPE);
                ftpClient.makeDirectory(file.getUploadedBy().replace(" ", "_"));
                String filePath = file.getFilePath() + ".pdf";
                file.setFilePath(filePath);
                done = ftpClient.storeFile(file.getFilePath(), inputstream);
                if (done) {
                    em.persist(file);
                }
            } catch (FileNotFoundException e) {
                System.out.println("File not file...");
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                try {
                    inputstream.close();
                    ftpClient.disconnect();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    } else {
        System.out.println("Could not connect to the ftp server");
    }
    return done;
}
```

---

### C. Polling Service Route - Maternity System to Birth Certificate System

```
public class RouteManager extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("timer:myTime?period=1500000")
            .process(new Infosource.messaging.rest.PrepareRestProcessor())
            .to("cxfrs:bean:rsClient").marshal().json()
            .process(new PrepareSoapProcessor())
            .to("cxf:bean:soapServiceHA");

        // from("quartz2://collectReady/scheduler?cron=0/15+*+*?")
        from("timer://collectReady?period=1500000").process(new Processor() {
            @Override
            public void process(Exchange exchange) throws Exception {
                MessageContentsList msg = new MessageContentsList();
                exchange.getOut().setHeader("operationName", "approved");
                exchange.getOut().setBody(msg);
            }
        }).to("cxf:bean:homeAffairs").marshal().json()
            .to("activemq:queue:GrantApplication");
    }
}
```

### D. User Profile Integration Service – Endpoint Definition

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:camel="http://camel.apache.org/schema/blueprint"
    xmlns:cxf="http://camel.apache.org/schema/blueprint/cxf"
    xmlns:cm="http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0"
    xsi:schemaLocation="
        http://camel.apache.org/schema/blueprint
        http://camel.apache.org/schema/blueprint/camel-blueprint.xsd
        http://camel.apache.org/schema/blueprint/cxf
        http://camel.apache.org/schema/blueprint/cxf/camel-cxf.xsd
        http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0
        http://aries.apache.org/schemas/blueprint-cm/blueprint-cm-1.0.0.xsd
        http://www.osgi.org/xmlns/blueprint/v1.0.0
        https://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">
    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
        <routeBuilder ref="routeBuilder" />
    </camelContext>
    <cxf:rsClient id="MartEndpoint"
        address="http://localhost:8080/BirthTech/services/rest"
        serviceClass="infosource.upservice.rest.MartService">
```

```

        loggingFeatureEnabled="true">
        <cxfr:providers>
            <ref component-id="jsonProvider" />
            <ref component-id="cors" />
        </cxfr:providers>
    </cxfr:rsClient>
    <cxfr:cxfrEndpoint id="BCEndpoint"
        address="http://localhost:8080/BCManager/services/bcs"
wsdlURL="http://localhost:8080/BCManager/services/bcs?wsdl"
        serviceClass="birthcertificate.ews.soap.IBEService"
endpointName="ws:ibcservice"
        serviceName="ws:BEServiceService"
xmlns:ws="http://soap.ews.birthcertificate/">
        <cxfr:properties>
            <entry key="dataFormat" value="POJO"></entry>
        </cxfr:properties>
    </cxfr:cxfrEndpoint>
    <!-- Doc Man -->
    <cxfr:cxfrEndpoint id="DocMan"

        address="http://localhost:8080/DocManWeb/FileService/FileService"

        wsdlURL="http://localhost:8080/DocManWeb/FileService/FileService?wsd
1"
        serviceClass="docman.services.FileServiceRemote"
serviceName="ws:FileService"
        endpointName="ws:SOAPFileService"
xmlns:ws="http://services.docman/">
    </cxfr:cxfrEndpoint>
    <!-- User Account Manager System -->
    <cxfr:cxfrEndpoint id="UserGate"
        address="http://localhost:8080/userGate/AccountService"
wsdlURL="http://localhost:8080/userGate/AccountService?wsdl"
        serviceName="ws:AccountService"
endpointName="ws:AccountServicePort"
        serviceClass="useraccount.soap.services.PersonInterface"
xmlns:ws="http://services.soap.useraccount/">
        <cxfr:properties>
            <entry key="dataFormat" value="POJO"></entry>
        </cxfr:properties>
    </cxfr:cxfrEndpoint>
    <bean id="routeBuilder"
class="infosource.upservice.routes.RouteBuilder"></bean>
    <bean id="cors" class="infosource.upsclient.cors.CORSFilter"></bean>
    <bean id="jsonProvider"
class="org.codehaus.jackson.jaxrs.JacksonJsonProvider" />
    <bean id="activemq"
class="org.apache.activemq.camel.component.ActiveMQComponent">
        <property name="brokerURL" value="tcp://localhost:61616" />
        <property name="userName" value="admin" />
        <property name="password" value="admin" />
    </bean>
</blueprint>

```



## E. User Profile Integration Service – Route Definition

```
public class RouteBuilder extends org.apache.camel.builder.RouteBuilder {
    // SoapJaxbDataFormat fmt = new
    SoapJaxbDataFormat("docman.services");
    @Override
    public void configure() throws Exception {
        from("vm:userProfile")
            .choice()

            .when(header("operationName").isEqualTo("createaccount"))
                .process(new Processor() {
                    @Override
                    public void process(Exchange exc) throws
Exception {
                        exc.getOut()
                            .setHeader("operationName",
"createAccount");
                        MessageContentsList content = new
MessageContentsList();
                        content.add(exc.getIn().getBody(String.class));
                        exc.getOut().setBody(content);
                    }
                })
                .to("direct:createaccount")
                .when(header("profileMode").isEqualTo("full"))
                .process(new SetBodyProcessor())
                .to("direct:subRoutes")
                .when(header("profileMode").isEqualTo("docs"))
                .process(new DocManProcessor())
                .to("direct:DocManEndpoint")
                .process(new Processor() {
                    @Override
                    public void process(Exchange exc) throws
Exception {
                        JSONObject DocManListJson = new
JSONObject(exc.getIn()
                            .getBody(String.class));
                        JSONArray fileList = DocManListJson
                            .getJSONObject(
"org.apache.cxf.message.MessageContentsList")
                            .getJSONObject("list")
                            .getJSONArray("docman.services.FileInfo");
                        JSONObject response = new JSONObject();
                        response.append("documents", fileList);
                        exc.getOut().setBody(response.toString());
                    }
                })
                .when(header("profileMode").isEqualTo("basic"))
```

```

        .process(new
UAServiceProcessor()).to("direct:UAEndpoint")
        .process(new Processor() {
            @Override
            public void process(Exchange exc) throws
Exception {
                JSONObject personJson = new
JSONObject(exc.getIn())
                    .getBody(String.class));
                JSONObject person = personJson
                    .getJSONObject(
"org.apache.cxf.message.MessageContentsList")
                    .getJSONObject("list")
                    .getJSONObject(
"useraccount.soap.services.Person");
                JSONObject response = new JSONObject();
                response.append("personalDetails",
person);
                exc.getOut().setBody(response.toString());
            }
        }).otherwise().end().to("stream:out");

        from("direct:subRoutes").multicast()
            .aggregationStrategy(new AggregationStrategy() {
                @Override
                public Exchange aggregate(Exchange
oldExchange,
                    Exchange newExchange) {
                    if (oldExchange == null) {
                        return newExchange;
                    }
                    try {
                        JSONObject DocManListJson = new
JSONObject(
                            oldExchange.getIn().getBody(String.class));
                        JSONArray fileList =
DocManListJson
                            .getJSONObject(
"org.apache.cxf.message.MessageContentsList")
                            .getJSONObject("list")
                            .getJSONObject("list")
                            .getJSONArray("docman.services.FileInfo");
                        JSONObject personJson = new
JSONObject(newExchange
                            .getIn().getBody(String.class));

```

```

        JSONObject person = personJson
            .getJSONObject(

"org.apache.cxf.message.MessageContentsList")
            .getJSONObject("list")
            .getJSONObject(

"useraccount.soap.services.Person");
        JSONObject response = new
JSONObject();
        response.append("personalDetails",
person);
        response.append("documents",
fileList);

        oldExchange.getOut().setBody(response.toString());
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return oldExchange;
}
    }).to("direct:DocManEndpoint",
"direct:UAEndpoint");

        from("direct:DocManEndpoint").setHeader("operationName")

.simple("getFilesByOwner").to("cxf:bean:DocMan").marshal()
        .json();
        from("direct:UAEndpoint").setHeader("operationName")

.simple("findByUsername").to("cxf:bean:UserGate").marshal()
        .json();
        from("direct:createaccount").to("cxf:bean:UserGate")
        .process(new Processor() {

            @Override
            public void process(Exchange exc) throws
Exception {
                String response =
exc.getIn().getBody(String.class);
                exc.getIn().setHeader("content-Type",
"application/json");

                JSONObject jobject = new JSONObject();
                jobject.put("message", response);

                exc.getIn().setBody(jobject.toString());

            }
        });
    }
}

```

## F. Consent Form



Dear participant

You are kindly requested to partake in research study on behalf of University of Fort Hare, Department of Computer Science, by **Lizo Masikisiki** (Principal Investigator) and **Sikhumbuzo Ngwenya** (Supervisor).

**Topic: Development of a Context Sensitive Rural Development Application.**

### **Research Context**

This study forms part of an ongoing research that seeks to produce a prototype of an interoperable, customizable and context-sensitive rural development application that enables rural communities to interact with services providers and one another electronically. The concerned researchers take an understanding of the current state of rural development in South Africa and the range of service required by rural inhabitants as preliminary steps to achieve the main objective of this research. This study therefor seeks to gather information about the range of services that rural inhabitants require from government offices as well as any other urbanized service providers.

Please indicate with a tick on the boxes provided below if you agree/do not agree to take part in this study.

I agree to participate in this study

☐

I do not agree with the study

☐

If you have any questions or comments regarding the study, feel free to contact the principal investigator on: **Cell: 0788329139, Email: lmasikisiki@ufh.ac.za**

## G. Questionnaire Sample



1. 

Gender	Female	Male
2. 

Employment Status	Employed	Unemployed
3. 

Age Group	16-19	20-29	30-39	40-49	50-59	60-69	70-80
4. 

Education Qualification	None	Grade 12	Short Course Certificate	Diploma	Degree
5. Do you own a cellphone 

Yes		No	
-----	--	----	--

  - a. If **yes**, which type of a cellphone do you own? 

Feature Phone		Smartphone	
---------------	--	------------	--
6. If you were to buy a new phone, which type would you buy 

Feature Phone		Smartphone	
---------------	--	------------	--
7. Please indicate your computer literate level 

None		Basic Level		Intermediate		Advanced	
------	--	-------------	--	--------------	--	----------	--

  - a. If **none**, Would you need training in computers 

Yes		No	
-----	--	----	--
8. Can you be able to use internet from your cellphone or from a computer? 

Yes		No	
-----	--	----	--
9. How often do you go to town?
 

Almost every day	Once a Months	More than Once a Month	Quarterly
10. How much is the transport fair 

R
---
11. Other than shopping, what else do you go to town for?
 

Private Offices	Eskom Offices	Municipality Offices	Eskom Offices	Education Offices
Health Offices	Home Affairs	Agriculture	Job Application	

Other (Specify) \_\_\_\_\_
12. Do you ever come back from any of these offices unattended? 

Yes		No	
-----	--	----	--

  - a. If yes, please specify reason(s).....
13. Do you think communicating with these offices electronically could **minimize** costs? 

Yes		No	
-----	--	----	--

## **H. Proof Reading and Editing Certificate**

### **TO WHOM IT MAY CONCERN**

#### **LETTER OF ATTESTATION**

**I, Dr. K. E. Monyai, hereby certify that I received and edited the Masters Dissertation of Masikisiki Lizo, entitled “DESIGN AND DEVELOPMENT OF A RURAL DEVELOPMENT ENABLER APPLICATION”, 145 pages. Proposed corrections to be implemented by the author were 193.**

**Director/ Editor/ Educator**

**Dr. K. E. Monyai (Ph D)**

***PARLONS LA LANGUE – LET US SPEAK THE LANGUAGE cc 2005/072166/23***

**Tax Clearance Certificate Number: 0007/1/2014/0006506743**

Diplôme d'études de la langue française (Ministère de l'Education Nationale, Paris)

Post Graduate Diploma RE (Corpus Christi College, London)

B Th (Urbaniana, Rome), B Phil (University of Hull, England), M Th, Drs (UNISA),  
Ph D (NWU, Tlokwe-Potchefstroom Campus)

P.O. Fort Beaufort, 5720

Email: [drkemonyai@gmail.com](mailto:drkemonyai@gmail.com) Cell: 0607694731 Fax: 0866282812

Date: 02 June 2017

**Disclaimer: The editor is not responsible for the non-implementation of the proposed corrections in the final version of the dissertation/thesis.**

