Declaration

I, Siyabonga Sifiso Cebekhulu, declare that this dissertation presents my own research work and has not been submitted in any form to any other tertiary institution for another degree or diploma. All the material used as source of information has been appropriately acknowledged in the text.

……………………………………..

SS Cebekhulu (Mr)

Dedication

I dedicate this work to my daughter, Siphesihle Sibongakonke Cebekhulu.

Acknowledgements

I would like to express my gratitude to my supervisor, Prof M.O. Adigun, for the guidance and support in making this work a reality. I would also like to make special mention of Mr E. Jembere and Mr M.T. Nene for their continuous assistance in shaping up the presentation of this work and in the prototype implementation. I am very grateful to Telkom, THRIP and NRF, for giving me this opportunity to further my studies. A special thanks to my CAP cluster members, P.T. Cwele, N.M. Gumbi and T. Shezi, for the continuous assessment and support we have offered each another during the course of our studies. My thanks, also to the whole research family in our department for their support. Last, but not least, I would like to express my gratitude to Mr V.E. Conco, for being the friend who has helped me throughout this study by providing me with mental support, when needed.

Abstract

Small, Micro and Medium Enterprises (SMMEs) play a huge role in improving the economic growth of a community. In the area of Nongoma, where this study is concentrated, most SMMEs still use traditional methods, such as street vending, in order to penetrate the market. However, these fall short in exposing the business, compared to those using electronic business (e-business) technologies. E-business can provide significant benefits such as cost savings, quick execution of business transactions, market for SMMEs collaboration and promoting globalisation of business activities. This work introduces an e-business technology in the rural areas of Nongoma with the hope that it can provide many opportunities for the resource-constrained SMMEs of that region. One such resource is exposure to affordable and reliable Information and Communication Technology (ICT) infrastructures. By introducing an ICT infrastructure to these SMMEs, the aim is to expose SMMEs' business activities to a much broader market of consumers. This research was aimed at introducing a Grid-based Utility Infrastructure for SMMEs Enabling Technologies, (GUISET), prototype for enabling SMMEs business activities without them actually owning the infrastructure. This is in the hope of giving them an opportunity to explore the global market.

In this research, software development paradigms were compared based on the GUISET requirements. The requirements, formulated from the GUISET characteristics, are tailored towards an infrastructure for addressing service composition and dynamic service selection. These requirements were used to select a development paradigm deemed to be the most suitable development strategy for the GUISET infrastructure. The emergence of open-source solutions have become a 'hot' topic in software development and tool support in the last decade.

Its application in different fields, including Service Oriented Computing (SOC), led to open source software being the natural choice for prototyping GUISET infrastructure and the Nongoma Online Stores, (NOS). The GUISET infrastructure supports the NOS application through the enablement of, (1): the assembly of components and (2): dynamic binding to services used by the defined assembly in (1), to achieve its business goal. The completion of a business process by the NOS automatically validates the GUISET infrastructure as service composition and dynamicity have been achieved.

Table of Contents

List of Acronyms

AOP	Aspect Oriented Programming
CBD	Component Based Development
CBSE	Component Based Software Engineering
GUISET	Grid-based Utility Infrastructure for Small, Micro and Medium Enterprises Enabling Technologies
GuSCaDA	GUISET Service Composition and Dynamic Binding Architecture
ICT	Information and Communication Technology
IT	Information Technology
NOS	Nongoma Online Stores
NTAC	Nongoma Tourism, Arts and Crafts
OSGi	Open Service Gateway initiative
ROA	Resource Oriented Architecture
ROC	Resource Oriented Computing
SCA	Service Component Architecture
SMMEs	Small, Micro and Medium Enterprises
SOA	Service Oriented Architecture
SOC	Service Oriented Computing
WWW	World Wide Web

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Overview

The importance of Small, Micro and Medium Enterprises (SMMEs) is vastly being recognized all over the world because of its contribution to economic development. SMMEs constitute a significant portion of the business sector especially in developing countries. Some of these SMMEs trade in arts and crafts, traditional wear of local tribes, and even in providing accommodation for tourists (Migiro and Adigun, 2005; Adigun, 2007; Dalvit *et al*, 2007; Njeje, 2008). Recognising the importance of SMMEs, countries around the world have established some initiatives to promote and to sustain SMMEs, especially in rural areas (Migiro and Adigun, 2005; Maswera *et al*, 2008; Modimogale and Kroeze, 2011).

Promoting and maintaining of SMMEs remains one of the highest priorities of the South African government because of its potential benefits to the local economy (DTI; 2005). The benefits derived from promoting SMMEs, especially in rural areas can include an increase in employment, improved market share and increased trading. SMMEs have only been able to deliver most of these benefits through the traditional market arena, whereas established big enterprises also use Web technologies to perform the same business activities thereby speeding up the process of reaching the targeted customers. The Web is one of the most revolutionary technologies today; it has been changing the business environment and has had a dramatic impact on the future of electronic business (e-business). The ability to perform business activities online is, among other things, resulting in a huge increase in Internet usage, be it buying goods, making bookings, or accessing other services.

The advent of the Internet and World Wide Web (WWW) back in the early 1990s also brought about the e-business paradigm which has since been considered as the modern and new way of conducting business. Researchers such as: Dalvit *et al,* (2007), Akoumianakis

(2008), Sibiya *et al*, (2008), Njeje (2008), and Modimogale and Kroeze (2011) have identified the need for SMMEs to expose their business ventures to the e-business paradigm, thereby taking advantage of the global market. This is gained by conducting business activities online with possibly millions of Internet users as potential customers. E-business is no longer an alternative, it is an imperative. By making Information and Communication Technologies (ICT) implementation one of the essential parts of their businesses, SMMEs can possibly compete with big businesses for the market share. This can in turn allow SMMEs to achieve their goal of survival and growth in the market. E-business is fast gaining ground as an accepted business paradigm, through the usage of ICTs, as more and more businesses are implementing website functionality for performing their business activities over the Web (Njeje, 2008; Dyakalashe, 2009). In order for an e-business to be successful, there is a need for building and deploying an infrastructure supporting these business activities. Unfortunately for decades, development of software applications has mainly been developing applications from scratch, which has seen huge amounts of budgets exhausted, longer time-to-market, and difficulties in extension of functionality.

Software development efforts have seen an emergence of a culture focusing on integrating existing and new modules in the development of software applications (Breivold and Larsson; 2007). This approach of integrating modules achieves faster time-to-market, re-using of modules and moving beyond the point of tightly coupled environments. It allows accessing of functionality residing in geographically dispersed locations of any platform at the point of execution. Over the years in the field of Software Engineering, several approaches have been attempted of putting together and/or developing and deploying an infrastructure capable of addressing users' requirements. Software development paradigms and technologies can be used in the same field to provide a systematic approach to the realisation of the intended software development project. Among the different development approaches in practice, none has been found to be superior to the other; rather, their application is based on project requirements. Project requirements are usually the optimal selection criterion for the software development paradigm applied in a development process. There are many software development paradigms available in the software engineering field. These software development paradigms have differences and similarities in some aspects based on features they provide in the development process. In other words, some paradigms may differ in the approach they use in the realisation of a project or when comparison is done, based on how they perform when given a particular functionality. How easy or difficult it is to apply the paradigm in the development process is usually based on the support the development paradigm provides. There are software development paradigms that are well documented and have been applied repeatedly in the development processes, while there are also those that have been created in ad hoc situations to achieve a specific goal.

Among the available paradigms, the selection of a paradigm for this work, concentrated on paradigms that offer substantial documentation and have had their share in application realising software development projects as candidate paradigms. These paradigms include: Component Based Software Engineering (CBSE), Service Oriented Computing (SOC), Resource Oriented Computing (ROC), and Aspect Oriented Programming (AOP). The evolution of development approaches, aims at applying a systematic and disciplined approach to development in addition to building infrastructures evolvable over time. This is ideal for developing an e-business infrastructure, enabling business activities where components are re-used among different business processes of more multiple users. The business activities of a user, be it an SMME, are achievable through some combination of existing modules, giving a look and feel of ownership of an infrastructure rather than in the case of merely a virtual environment.

This work introduces a Grid-based Utility Infrastructure for SMMEs Enabling Technologies (GUISET) as an e-business solution for the Nongoma area. The concept of GUISET is further elaborated in Chapter 2 as the building block for this work, furthermore, an examination of several software development paradigms and their supporting tools for developing an ICT infrastructure for enabling SMMEs is conducted. This is done to assist SMMEs to expose their business activities online. Thus, the indications of the problem at hand are identified, which is how the infrastructure has been characterised. This is then followed by the identification of the operational characteristics the infrastructure must exhibit, which is how the basis of comparison for software development paradigms applied has been formed. Finally, the open-source solutions world is explored to deduce possible tools for the development of an e-business application serving as proof that the GUISET infrastructure has been realised.

1.2 Nongoma: The Area of Study

In most parts of the world there are regions deemed as underdeveloped, especially in developing countries in Africa and Latin America. These regions have formed SMMEs that produce various products with the hope of selling them mostly to passing tourists. SMMEs, especially in developing countries such as in Africa, Asia, and Latin America, lack infrastructural exposure to ICTs and operate using traditional tools (Adigun, 2007; Njeje, 2008; Dalvit *et al*, 2007). Various regions in South Africa organise local SMMEs into small cooperatives. This study focuses on the Nongoma Tourism, Arts and Crafts (NTAC) which embody various SMMEs in Nongoma (See an aerial view of the Nongoma area in Figure 1.1 below.) Several authors such as Migiro and Adigun (2005), Adigun *et al*, (2006), Dyakalashe (2009), Chiliya *et al*, (2011) state the adoption of ICT can give SMMEs the ability to be innovative and flexible, widen their market and extend their business hours.

For SMMEs to succeed in using ICT as a competitive tool, they should align ICTs with the business goals and integrate it with their business processes.

The Nongoma Tourism, Arts and Crafts (NTAC) SMMEs offer various products ranging from arts and crafts to traditional wear while others offer services such as accommodation for tourists. Most of these formations operate in the traditional market using traditional methods such as vending tables, street vending and other means, including standing by the street and even walking between lanes, where a seller carries his products on a box or by hand for display. These SMMEs lack the finance that is needed to own the ICT infrastructure, such as computing resources that are needed to operate an online store thereby taking opportunity of the e-business paradigm. Herselman (2003) stated that South Africa is developing in an uneven manner with the urban areas having the modern technologies, whilst rural areas are having little or no infrastructure needed for venturing into the e-business world at all. Even though this might be a shortcoming to the inhabitants of rural areas, access to ICT can be leveraged through the usage of cell phones which most South Africans now own. Cellular phones are a good platform for delivering services.

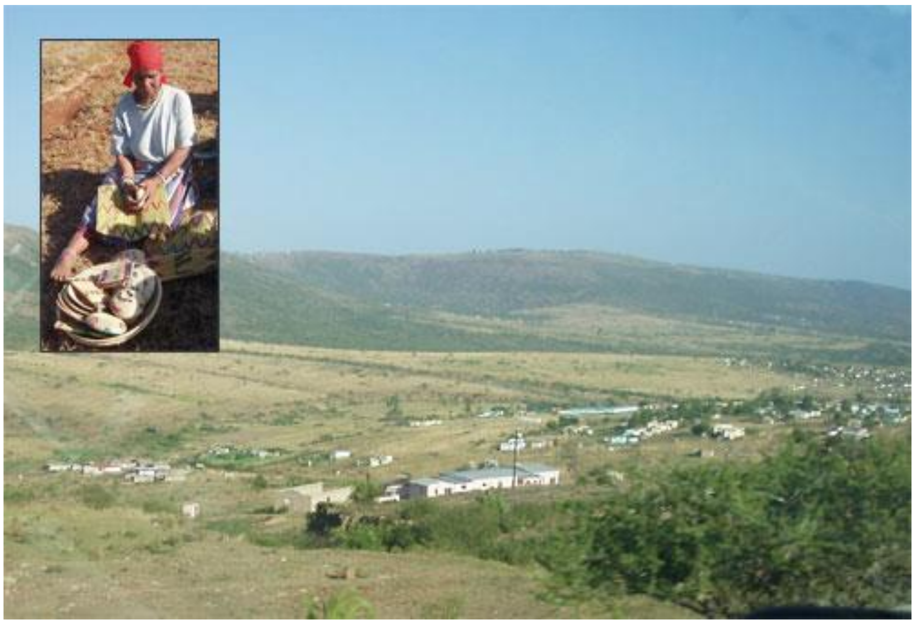


Figure1.1: An Arial view of Nongoma Community. Insert: A Nongoma woman with her arts

Over the recent years Nongoma has been increasingly becoming a tourist attraction destination mainly because of its housing of the royal palaces which belong to the current Zulu king, King Goodwill Zwelithini. There is also an annual event called "uMkhosi wo Mhlanga" (the reed dance) attracting thousands of tourists. The popularity of the reed dance which is a Zulu tradition, has so much attraction people of many races come to watch it. The family members of the royal family, as a way of promoting interest in Nongoma, are often willing to speak to tourists about the Zulu Royal House and the traditions of its people. They also allow tourists to visit their homes, (Zululand Tourism, 2009). Dr Liz Nyholm of the Omnia Practice in Yardley Green Medical Centre, talks of her experiences at the reed dance in her blog and some comments posted on the blog are appealing to possible/future tourists yet to visit (Nyholm, 2010). The growth in tourists visiting, in turn, provides opportunities for the SMMEs in trade to display their products. This exposure of Nongoma makes it a noticeable place especially showcasing its traditional wear, the area strongly perceived to be the motherland of the Zulus. During these events products are greatly exposed to tourists, but the drawback is that these products are available throughout the year while events are only held once in a year.

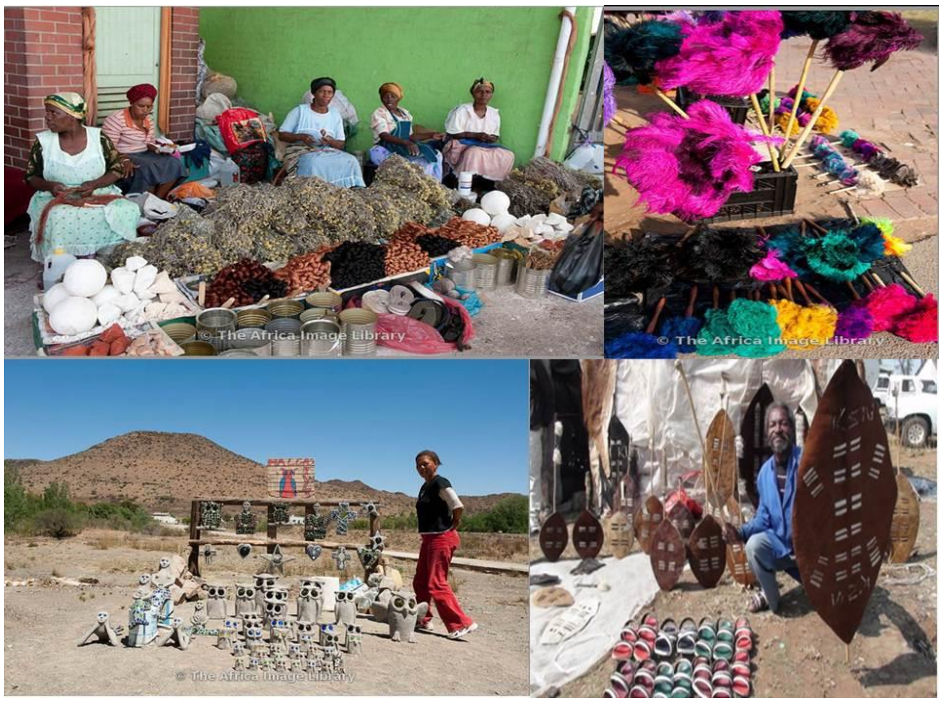

© The Africa Image Library
© The Africa Image Library
© The Africa Image Library

Figure1.2: A typical market for products (pictures from The Africa Image Library: http://www.africaimagelibrary.com/)

Figure1.2 shows a typical example of how business is conducted in the traditional market where sellers (SMMEs) start trading anywhere, on the street, outside malls, standing by traffic lights. Most of these forms of trading are illegal. There is thus a need to develop an IT infrastructure to enable the NTAC to offer to its embodied SMMEs. In this way, their products will be available to their national customers throughout the year. At the time of conducting this research, there was no existing electronic enabling infrastructure for the SMMEs in the region of Nongoma. Having an e-business enabling infrastructure gives greater advantage as Sun (2011) stated that one of the most popular activities conducted on the Internet is through online shopping. As he clearly states: "it can be done in your leisure time, even in one's pyjamas".

1.3 Indications of the Problem

In the early stages of this research, the focus was on studying the adoption of ICT among SMMEs in developing countries as an e-business enabler. This effort revealed diverse shortcomings hindering the growth in the usage of ICTs to conduct business activities. Issues negatively affecting the growth of e-business adoption by SMMEs include: lack of Information Technology (IT) knowledge; cost of IT; lack of understanding of the benefits e-business enabling technologies can provide; and how to measure those benefits (Management Services, 1997; Igbaria *et al*, 1997; Pollard and Hayne, 1998; Adigun *et al*, 2006; COFISA Project Plan, 2008; Cloete *et al*, 2002; Kshetri, 2007). Most SMMEs are still using the traditional methods to survive. By traditional, reference is made to street vending and cluster of tables for selling at taxi/bus stations. What they need is to take advantage of the power of ICT in order to be competitive whether small, big or global scale (Modimogale and Kroeze, 2011).

GUISET aims to be a starting point for SMMEs wanting to join the supply chain linkage with large enterprises. Opportunities such as employment, poverty alleviation, as well as social growth are likely to increase by the development of an infrastructure to support SMMEs in rural communities (Adigun *et al*, 2006, Warden and Motjolopane, 2007; Golding *et al*, 2008, Ekabua, 2010).

In order to realise an infrastructure for enabling business activities to be conducted electronically, there is a need for collaboration among the necessary technological tools. Composition of services for creating environments supporting business-to-business process or enterprise application integration has attracted a lot of attention over the years (Bucchiarone and Gnesi, 2006). Li and Parashar (2006) state having available services may not necessarily, in itself, address an application's requirements but rather several services need to be composed in an application-specific manner to construct the functionalities required. Currently, the state of software applications shows that there is a lot of dynamicity involved. The aim is building software applications that require less or no human intervention. Zhou and Niu (2010) emphasised how critical it is for an e-business infrastructure to be dynamic and upgradable at runtime to increase flexibility. If there is an application that has to be developed it has to dynamically get what it is going to use in a given case to satisfy its business goal.

Among the constraints hindering the adoption of ICT by SMMEs is the cost of owning the infrastructure. This was the preliminary motivator for selecting open-solutions in the development of the infrastructure, which later prompted motivators such as availability, access to code and possibilities of extension to satisfy specific goals of the developer. Pastrana and Lopez (2009) state evolution has been gained by open-source solutions in the development of software also avoiding vendor lock-in, and together with Jarvensivu *et al* (2006) they state the possibilities of integrating several open-source solutions into one software application. Yan *et al* (2009) stated that finding a suitable approach of combining existing tools to realise an infrastructure has proven to be a challenge due to the diversity of business functions and accommodating legacy systems.

Given the possibility that existing technologies could be used to compose an infrastructure for e-business support; a strategy is required that, (i), identifies the candidate technologies as well as, (ii), combining them to realise service composition and dynamic service binding. The Centre for Mobile E-services for Development at the University of Zululand is currently working on a research theme called GUISET (Grid-based Utility Infrastructure for SMMEs Enabling Technologies). GUISET is an architecture targeted at providing IT services such as application software to SMMEs without actually owning them, (Sibiya *et al*, 2008; Ekabua, 2009), to conduct business activities.

GUISET proposes to be a technology enabler that fits into the service oriented arena by being both the service producer and service consumer. Several research activities have been explored and reported at the centre addressing some aspect of GUISET and this work is also one of them. In the previous attempts to realising GUISET, none has concentrated on producing a prototype that mainly addresses issues concerned with SMMEs. Since GUISET is a portal of services offered to SMMEs, this work specifically concentrates on finding a strategy for composing and dynamically binding services for its service consumers to achieve their business goal.

1.4 Statement of the Problem:

In the past software development depended on building solutions from scratch. Presently most enterprises are trying to move away from that approach towards the approach of building by integrating existing infrastructures. Adigun *et al*, (2006) stated that there is an affordable technology for SMMEs, all that is needed is an appropriate strategy to make these technologies available using the utility approach to service delivery. The GUISET reference model is yet to be realised in a concrete architecture. In this work an argument is presented on how service composition is at the heart of GUISET. Therefore putting together a combination of existing technologies in a GUISET-style prototype can be used to demonstrate GUISET enabling of basic e-business.

1.5 Research Question(s):

To address the issue of developing an ICT infrastructure that supports service composition and dynamic service binding the following question is addressed in response to the gaps defined in the GUISET reference model:

1. Which mechanism can be used to demonstrate the GUISET Software Infrastructure?

GUISET will be demonstrated by deploying an infrastructure aimed at assisting SMMEs in Nongoma to conduct their business activities online. The proposed strategy addresses the following sub-questions to qualify as GUISET:

- How can service components be coordinated to realise a specific e-business goal?
- How can dynamic integration of components be achieved?

1.6 Rationale of the Study:

The expected outcome of this study is a software infrastructure that is evolvable in with time to provide for additional functionalities to be performed. By enabling service composition and dynamic service binding support for the infrastructure there are possibilities on cost savings in the development process of the infrastructure. This, for the resource constrained SMMEs, is a benefit as an application is assembled on request with quick time to market. A Nongoma Online Store is deployed to demonstrate addressing some aspects of the infrastructure. The prototype enables cooperatives to register and sell their products online. The infrastructure is aimed at having re-usable business services, allowing business activities to be realisable from a set of services with a service possibly being used in a number of activities at once. Software developers will gain foundational knowledge for selecting a suitable paradigm for deploying an e-business infrastructure using open source solutions.

1.7 Research Goal and Objectives:

Research Goal:

The aim of this research is determine the minimum functionalities that configure the GUISET architecture reasonably.

Objectives:

To achieve the goal of this research, the following objectives are envisaged:

i. To identify the basic functionalities of GUISET to be interfaced.

ii. Do a comparative analysis of candidate development approaches/paradigms to generate possible GUISET configuration alternatives.

iii. Use the basic functionalities to identify candidate open-source tools guided by results of objective (ii) that can be used to craft GUISET.

iv. Demonstrate at least one GUISET configuration.

1.8 Project Delineation:

The work covered in this research is limited to realising the first GUISET configuration that supports dynamic service binding. The application of evolutionary prototyping was mainly for developing a prototype that can go through a number of iterations until it fully satisfies the GUISET requirements, easily extendable with additional functionality as the number of requirements increases. For the purposes of this research only a single iteration of the chosen configuration was developed with the intentions of showing how using SOC technologies GUISET requirements can be met. This work forms part of an on-going research by the Centre of Excellence for Mobile e-Service, Department of Computer Science, University of Zululand, South Africa, which aims at building an evolutionary infrastructure, the GUISET. The scope of the work reported in this document was limited to finding an architectural configuration together with its implementation tools that can be used to demonstrate that the GUISET infrastructure supports dynamic service binding, as the core requirements of GUISET are service composition and dynamic service binding. Due to the configuration of GUISET requirements discussed, this then limited the study to involve no user evaluation. The exclusion of users is based on the fact that the issues of dynamic service binding are deemed to be technical issues that are not easily contributed to by basic users; the expectation is that they are limited to the development team.

The chosen configuration is an e-business infrastructure that holds and acquires services to be used by its member groups. These services are then composed on demand to fulfil some business goal as expected by the group. Therefore, the GUISET configuration under consideration, is only limited to achieving service composition and dynamic binding of services in a typical business process. The validation approach for the business process relied on completing one or two business transactions, which in this case involved searching the product catalogue and placing an order for the product of choice. In this instance some of the services that are composed worth mentioning are the catalogue service, order service, warehouse service and the email notification service and these services dynamically bind to resources to ensure the right provider is chosen to satisfy the request. Upon the success of the mentioned validation approach which was executed using the Nongoma Online Stores (NOS) an e-business application, then the GUISET configuration requirements can be said to have been met. Using the NOS, a number of services are composed to complete the business process of placing an order for a product. The NOS itself is an approach that has been adopted for the purposes of evaluating the configuration.

1.9 Research Methodology:

In the process of conducting this research work, a number of research methods were applied. The research methods used are presented here to give a clear indication on how they have been applied in the process of conducting the research.

1.9.1 Literature Survey:

Literature was searched in the following knowledge areas: GUISET prototyping requirements; survey of the software development paradigms to match GUISET prototyping requirements; existing technologies – support for service composition and dynamic service binding. The areas explored in literature involve understanding the nature of e-business adoption in rural communities, both in South Africa and abroad, as a close relation since there is not much reported about the area of study. Software development paradigms such as SOC, CBSE, etc., and their supporting tools, especially open-source solutions, were explored to select one as an approach to the deployment of the desired infrastructure.

1.9.2 Prototyping:

A prototype implementation was put together using the tools selected when the paradigms were compared based on the GUISET requirements. The prototypes, the GUISET infrastructure realised through a combination of existing open-source tools and the Nongoma Online Stores serving the purpose of proving the GUISET infrastructure has been realised. Both these prototypes are implemented using the Java programming language particularly for the implementation of business logic and other tools such as Apache Tuscany, Eclipse Equinox and others. The use of open source tools was deemed as an important endeavour when the prototypes were implemented with opportunities to swap and change between varieties of vendor-specific products with no costs involved. The Nongoma Online Stores (NOS) aims to show how an e-business application can be deployed using Service Component Architecture (SCA) to enable the ease of using services implemented in any technology and also addressing some of the underlying issues in research such as in-service composition and dynamic binding to services to deploy the infrastructure using dynamic modules of the OSGi environment.

1.9.4 Proof of Concept:

To qualify the service-oriented application as a viable GUISET configuration, the NOS application demonstrates how the configuration assembles components and using services on the fly. The NOS application configures a business process on behalf of an SMME through the assembly of business components at design time and in turn uses services on the fly to satisfy the business goal of these business components. A successful completion of the business process validates this configuration as the GUISET infrastructure. This is what has been deemed sufficient for demonstrating that a typical GUISET configuration is proved feasible. The NOS is just one typical configuration f GUISET showing what can be achieved from combining business services based on the intended requirements that must be fulfilled.

1.10 Organisation of the Thesis:

The rest of this dissertation is briefly organised as follows:

Chapter 2: The GUISET Infrastructure: The chapter seeks to lay out the background knowledge of the GUISET concept using both grid and utility computing as its building blocks. Then the existing research challenges in service composition and dynamic service binding presented show how GUISET can be a realisable software development artefact using the available technologies. Then a motivating scenario and the infrastructural characteristics are presented to outline what support is provided by GUISET in the context of this work.

Chapter 3: Software Development Paradigms: The chapter begins by presenting the GUISET based evaluation framework stating the requirements to be met for an infrastructure to qualify as GUISET. This is then followed by the introduction of the software development paradigms that were candidates for this project. Lastly a comparison of these software development paradigms was conducted to find one paradigm that can be used to realise the GUISET infrastructure.

Chapter 4: Open-Source Solutions in GUISET infrastructure:

The chapter presents the conceptual architecture informing the implementation details of the infrastructure and the concept of software development and evolution together with the approaches that can achieve it. The chapter then presents the open-source solutions that have been adopted for the development of the infrastructure to enable building applications evolvable over time. The open-source solutions selection is based on the approaches discussed in the software development and evolution section.

Chapter 5: GUISET Infrastructure Realisation:

The chapter started by presenting the assumptions made on the prototypes; the GUISET infrastructure and the NOS application. This is then followed by the prototyping concepts acting as guidance for the type of prototyping conducted in this work. Lastly, the prototype implementation of NOS, a typical application supported by the GUISET infrastructure, is presented as proof-of-concept that the GUISET infrastructure has been realised.

Chapter 6: Conclusion and Future Work:

The chapter started by giving a brief introduction to the aims of this work; furthermore, a summary of the activities performed conducting this research is given. Then the limitations and future work are discussed, with the limitations outlining the intentions of this report and future work giving directions on the suggested future directions for extending this work.

1.11 Chapter Summary:

The chapter started by presenting foundational information on the research already that has been conducted with the aim of sustaining SMMEs in rural areas using Information and Communications Technologies (ICT), e-business development and the software development paradigms which this work is based upon, followed by some background on Nongoma which is the area being studied to clarify the significance of providing an e-business infrastructure in that region. This is then followed by indicators of the problem in providing the GUISET infrastructure. The GUISET infrastructure is thoroughly explored in the next chapter as the building block for this work.

Chapter 2: The GUISET Infrastructure

2.1 Introduction

This chapter introduces the software development concept in web applications as an approach to realising the grid based GUISET infrastructure. Grid is defined by Ekabua (2009) as a collaboration of geographically dispersed organisations with the intention of sharing physically distributed resources virtually as a single resource for its users. Grid computing enables the provisioning, accessing, sharing and management of resources and services which can be application software, storage facilities, processing capabilities and databases. The vision of grid computing is bringing together heterogeneous resources and allocating them efficiently to applications.

GUISET was proposed by Adigun *et al,* (2006) to provide capabilities of both grid and utility computing in a single environment to SMMEs. The primary aim of GUISET is to provide a service market for SMMEs. It offers a special intermediary for service discovery, selection, negotiation and guaranteed quality of service on a utility-based approach. Access to GUISET is acquired through group memberships that acquire a minimum set of native services for use by its members. Members of a group, among others, are SMMEs that have formed a cooperative relationship prior to joining or who gain membership by joining one of the existing groups. All the GUISET native and acquired services are stored in a repository. The utility approach is envisioned to provide an affordable access to Grid services deployed in the infrastructure. An SMME is only charged by the group for business services composed together to complete a business process. The application of a pay-per-use charging model is meant to address the lack of financial stability constraint. This is for enabling SMMEs to operate swiftly in the e-business market as costs are varied towards usage.

2.2 GUISET: The Foundational Reference Technology:

The GUISET architecture is based on the idea of putting together existing computing resources and/or technologies in the realisation of the GUISET infrastructure. This section gives an overview of the concepts of Grid and Utility Computing as the foundation and motivating force behind the GUISET infrastructure. This section has been organised as follows: Section 2.2.1 discusses the concepts of grid and utility computing as the main building blocks in the realisation of the infrastructure; Section 2.2.2 briefly outlines the GUISET architecture and gives an in-depth look at the different layers of the architecture and Section 2.2.3 discusses the research challenges to be met in the realisation of the GUISET infrastructure.

2.2.1 Grid and Utility

The purpose of Grid was to allow access to geographically-dispersed and underused computing resources to provide the required computing capability. A "resource" in the context of grid computing refers to any entity that can be used to fulfil the requirements of users. Foster *et al* (2001) states that sharing in grid computing goes beyond that of file sharing, but rather their main concern is on sharing resources. In that way a resource can be a computing node, data storage and/or software applications. Even though grid computing was introduced by research organisations to support computer-intensive scientific applications and share, at most, massive research databases, the latest developments in grid computing have seen the concept recognised as a foundation for flexible management of internal IT resources by all types of organisations (Srinivasan and Treadwell, 2005). The organisations that have adopted the usage of grid computing and closely related to what GUISET aims to achieve to mention just a few are SAS (http://www.sas.com/en_us/home.html), TIBCO Software Inc (http://www.tibco.com/), IBM (http://www.ibm.com/us/en/) and Hewlett Packard (www.hp.com/go/grid). The aforementioned enterprises have adopted the use of business grids they have developed to assist in providing needed platforms for addressing business needs such as e-commerce platforms. The intention of business grids is mainly to address the ever-changing needs of e-business technologies. A perception of increased business value is enough for an organisation to adapt to using business grids. Business grids middleware are responsible for enabling dynamic allocation of IT resources to business applications (Savva *et al,* 2004). A distributed system aiming to dynamically aggregate and co-ordinate various resources across enterprises and improve their utilisation in a way that improves or increases the overall productivity that can be loosely defined as an enterprise grid (Nadiminti and Buyya, 2005). Enterprise grids basically entail using grid computing within the context of a business or enterprise instead of using it in solving scientific problems. Enterprise grid is a kind of grid computing approach that can make grid computing an attractive approach in the business setting.

Grid computing aims to promote full utilisation of resources that have been deemed under-utilised by offering them for usage, whenever in idle mode, to other users. The main objective of Grid computing is resource sharing and coordinated problem solving in dynamic and multi-institutional virtual organisations (Foster *et al*, 2001). A computer grid can be provisioned on demand to support a variety of enterprise applications and users in turn align their IT demand according to their business activities. It is worth noting that grid computing technologies still need some improvements to match other computing technologies so as to truly achieve distributed resource sharing across heterogeneous and dynamic environments. True distributed resource sharing enables easy integration between several resources available for utilisation. The computer grid can also be viewed as virtual organisations (VO), as they are mainly formed to solve ad hoc problems with their existence usually relying on the duration of the problem being addressed. As Joseph *et al* (2004) stated, the success of grid computing clearly depends on integration and service orientation by creating these VO that use some combination of services to solve specific problems.

Grid computing lays the foundation for adaptive enterprises and the vision of utility computing by providing collaborative high-performance computing resources and data sharing. In an adaptive enterprise, goods' or services' demand and supply are matched and synchronised at all times. Such an organisation optimises the use of its resources, (including its information technology resources), always using only those it needs and paying only for what it uses, yet ensuring that the supply is adequate to meet demand. In order to achieve the goals of grid computing, the utility computing model has been suggested for deploying grids in an enterprise (Savva *et al*, 2004). Web services are the core mechanism for both grid and utility computing as they provide the functionality required. Web services enable ease of applying the utility model as they possess the capability of offering the necessary functionality without any additional ties such as owning the service.

Utility Computing as a service provisioning model that enables a service provider to make available computing resources and infrastructure to service users on request. This model takes the same mechanism as that of utilising electricity, where it encompasses the ideas of outsourcing and on-demand availability. The key concept of utility computing is outsourcing. In utility computing services offered charge on pay-as-you-go basis thereby enabling users to pay per use. This model can have an added advantage towards minimizing initial costs or no costs at all in the acquisition of computing resources. The service user has the ability to connect to services made available by several service providers at the time of execution. Using the utility model these are the potential benefits an organisation might benefit from: reduced fixed costs, treating IT as a variable cost, unlimited access to computing resources and improved flexibility, thus making the provision of services more agile and adaptive (Llorente *et al*, 2006). In GUISET, a group, in addition to owning a set of services, can acquire additional services in the market based on the need; these services are then shared between the members SMMEs.

The application of the utility model billing can benefit SMMEs with greater flexibility in costs as paying will only be for the resources used. SMMEs as stated by Adigun (2006) and Pastrana (2009) are too financially constrained to own a software infrastructure. They need a reduced investment mechanism on software to expose their business activities using e-business technologies. The benefits of using utility computing can be described as the controlling of costs, the improvement of service quality and efficient business responsiveness. With utility computing in place SMMEs can concentrate on their core business activities. The assumption is because SMMEs resources are limited, adoption of utility computing is a great benefit in that they only pay-per-use from the limited resources. This utility approach alleviates unnecessary costs incurred from a standardised fee. In fact idle grid resources can be made available to SMMEs in order to reduce costs. This makes it possible for resources prices to be negotiable.

The GUISET infrastructure aims to provide a one stop reservoir of services that can be accessed to address the issues of on-line presence for SMMEs. Adigun *et al,* (2006) states that GUISET is based on the idea that there is an affordable technology for SMMEs. All that is needed is an appropriate strategy to make these technologies available using the utility approach to service delivery. This research investigates how technologies supporting SMMEs business activities can be made available for easy access. An SMME is only exposed to a GUISET portal to perform its business activities and is not bothered by the technicalities involved in the successful execution of that business activity. The approach taken relates to addressing SMMEs' IT enablement related issues such as: reducing operating overhead, transforming to an e-business as a priority and the capability of using e-business tools without owning them.

2.2.2 The GUISET Architecture

The GUISET architecture, see Figure 2.1, is a layered architecture that is divided into three layers. A layered architecture gives an abstract view of the infrastructure and makes it possible all layers to be developed independently of others. It also shows which layer influences or interacts with which layer. The GUISET infrastructure layers are: (1) Multi-Modal Interfaces, (2) Middleware Layer and (3) Grid Infrastructure Layer. The architecture shows the relationship among the entities in the environment with each layer providing the necessary resources to be used by the layer above it in the realisation of the GUISET infrastructure. A short description of each of the three layers illustrated in the architecture is given below:

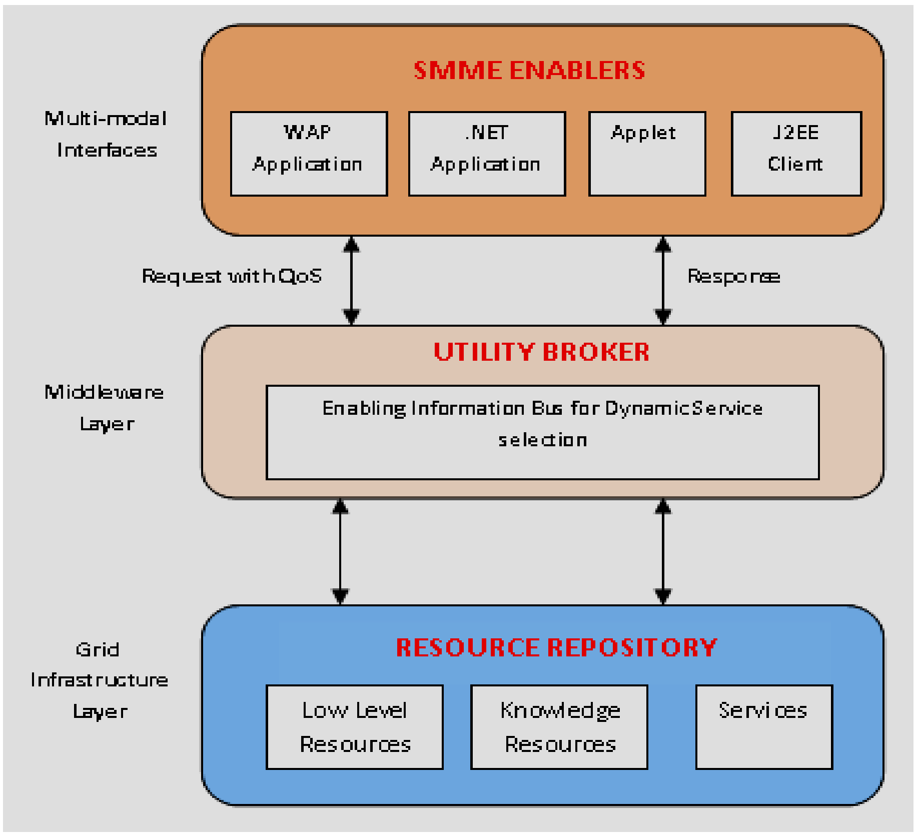


Figure 2. 1 Grid-based Utility Infrastructure for Small, Micro and Medium Enterprises

Enabling Technologies (GUISET) architecture [Adigun *et al,* (2006)]

1. Multi-Modal Interface

   The Multi-Modal Interface provides mechanisms that enable the interaction of service consumers with the infrastructure. The GUISET portal is a gateway to the GUISET native services and services acquired from the market. It establishes communication (by accepting requests and giving responses to users of the infrastructure) with the Middleware layer to fulfil its business goals. The middleware layer is responsible for the combinations between technologies and the composing of services taking place to achieve any process.

2. Middleware Layer

   The Middleware layer provides the business logic that is needed to support service provisioning by the Grid Infrastructure Layer. This layer using infrastructure rules, supports the consumption of grid resources in one of the following ways, simple sharing of a grid resource with other consumers; re-using a service by becoming an additional consumer; or joining a queue of users waiting for a resource until such a time when the resource is cheap enough for the consumers. The core of most research that has been conducted in an attempt to realise the GUISET concept has been focusing on this layer, among the dissertations or thesis worth mentioning are Sibiya;s masters dissetation, 2009; Iyilade's doctoral thesis, 2010; Mathonsi's master's dissertation, 2011 with their aims at improving the service provisioning capabilities.

3. Grid Infrastructure Layer

> The Grid Infrastructure layer is the storage for all the GUISET resources. It is a repository for both the native services and the services that were acquired from the service market. Grid resources are managed using grid middleware. The resources provided by the GUISET infrastructure range from data storage, software applications, services and technologies in the assembly of a business process satisfying some user goal. These services are responsible for day-to-day business activities of the infrastructure. Request for a typical resource therefore simply invokes the relevant grid service.

Most of the research on the GUISET concept has focused on the Middleware Layer. This is where mechanisms for interaction with the Grid Infrastructure Layer are addressed. Iyilade *et al,* (2009) proposed MINDS a middleware infrastructure that aims to perform dynamic composition of services in a grid environment. A middleware infrastructure based on software agents as proposed by MINDS is able to address composition issues where grid services possess the capability autonomy, heterogeneity, flexibility, and robustness. The Grid Infrastructure Layer is a repository for services offered by the infrastructure. To realise the GUISET infrastructure focus has been on assembling tools based on the architecture in Figure 2.1.

In light of the foregoing, the aim of this work is to prototype the GUISET infrastructure using available tools. These tools should address specific issues, namely; service composition and dynamicity which GUISET supports. In GUISET, as a distributed system, it is possible to deploy software applications as published services which consumers can subscribe to for performing their business activities. The business activities are provided through a collaboration of both service providers and service consumers using some communication protocol. The communication between the service provider and its service consumers is established via a network of brokers exchanging data. A set of archetypes in Figure 2.2 were crafted in order to simplify the architectural framework. There are three archetypes of components in the GUISET architecture. First, is the *Technology* archetype which refers to the set of service producers in the network. The Second archetype is called a *Client*, referring to the devices on which the services are deployed. Third is *Services* produced for Clients to consume which could be of type Information Service, Transaction Service, or Third party Service.

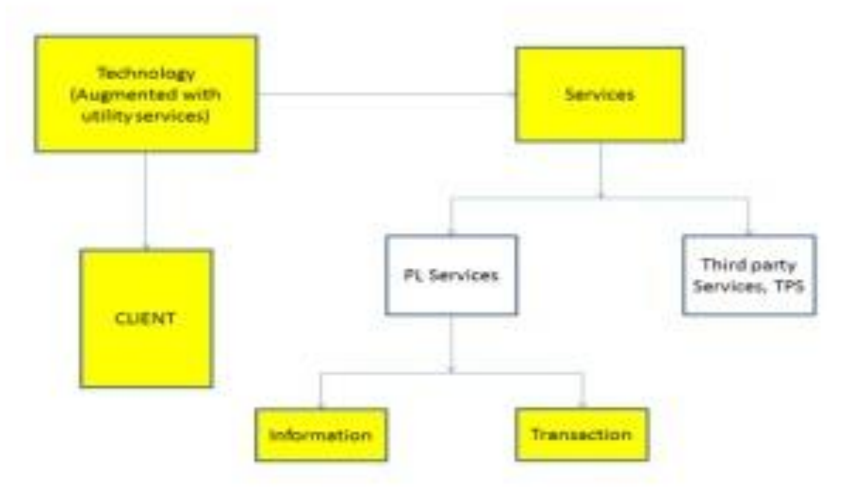

Technology (Augmented with utility services)
Services
CLIENT
PL Services
Third party Services, TPS
Information
Transaction

Figure 2.2: Reference Architecture for the Mobile Commerce Product Line [Adigun *et al*, (2006)]

2.2.3 Research Challenges in GUISET:

GUISET has a number of challenges that it aims to address. For the purposes of realising the GUISET software infrastructure using available technologies, the following GUISET challenges were addressed:

(1) Service Provisioning in a dynamic environment:

The main goal of the reference model is facilitating a loosely coupled implementation of networked services in an event-driven architecture. Services are of the pay-per-use type and are dispatched by a network of brokers to deal with context event notifications.

Services should be easily adaptable and scalable to satisfy the goals of the infrastructure. The GUISET infrastructure, as proposed, intends supporting the assembly of business services and dynamic service binding for enabling applications running on it to successfully execute. The infrastructure defines the business process at design time using interfaces. Services providing concrete implementation are discovered dynamically rather than being created by the consumer. Distributed service provisioning requires methodologies allowing end users to construct composite applications based on basic operations such as discovery and interconnecting remote sources. Services supporting the basic operation of the infrastructure can be implemented in any technology. In an environment where services must be dynamically composed, performance usually becomes an issue which can lead to no guarantees in service provisioning. A programming model that is technology agnostic will improve the performance of the infrastructure by establishing communication with a variety of services available to enable the completion of a business process. The idea is to pull together services from different providers and combine them in an effort to complete the business process of the requestor, an agent acting on behalf of a client who is performing its business activities and only paying per use e.g. a number of services from different providers may be used in fulfilling one client request. The implementation details and location should not matter but rather that these services are combined to achieve the goals of the consumer.

(2) Service Composition:

The GUISET infrastructure follows the notion that services are available; all that is needed is a mechanism that facilitates access to these services to satisfy some business activity. As an already mentioned requirement, a dynamic environment is needed to facilitate the interaction of services with the ease of adoption and scalability of distributed services. Some atomic services can be enough to build a software application. Mostly in practice, though, the atomic services are composed to provide some desired holistic functionality. Since services are fast becoming the major building block for software functionality, it is essential that a service composition strategy that supports both services and legacy applications implemented in some technology is used. Services are distributed and so may be sourced internally or externally. Typically, approaches to service composition can be data oriented, process oriented, transactional or a combination of these but none of these approaches are flexible enough to support all characteristics of service composition (Yang *et al*, 2002). Several infrastructural requirements hinder the success of the service composition approaches. To minimise the hindrances brought by the infrastructural requirements, this work looks at an approach that concentrates on the business logic as the main goal of the infrastructure. This gives opportunity to addressing the functional requirements of the infrastructure. For an infrastructure to qualify as having applied service composition, these are the steps that need to be exhibited:

- Creation of a process model specifying control and the data flow among the activities;
- Discovery of concrete services that complete the business process must be bound;
- Potential clients must be granted access to composite services; and
- On invocation of a composite service a coordinating entity may manage data flow on the basis of the associated process flow.

Automation of the outlined steps has recently been receiving a lot of attention from researchers. Iyilade *et al* (2009) defined dynamic composition as a task of selecting atomic services in a process without the interference of the service requestor. This is an ideal situation in e-business solutions as a dynamic environment is enabled. A dynamic environment facilitates the interaction of services to complete a process at execution time.

2.2.4 Gaps Left by the Reference Architecture:

The information gathered from the reference model enables or builds an understanding of what needs to be put together as a family of applications with the intention of investigating dynamic composition. The reference model does not provide a detailed model for service composition which in turn leaves a number of issues not addressed. Among the problems that have not been addressed in previous attempts to realise GUISET is how the service components will be coordinated. What communication protocol is going to be used for the services to communicate and the interfacing of services to guarantee that the quality of services is guaranteed? While it is known that SOA and CBSE provide a general framework for creating product family members (Adigun *et al,* 2006, Sibiya *et al,* 2008, Ekabua 2009), definite answers to following questions are still needed: What sort of applications will the members be? Can composite applications be composed from other applications? How will sharing of applications be supported? This research contributes in addressing these questions by realising the GUISET infrastructure.

Some related works that have been conducted in order to address GUISET related problems include (Chani, 2008) and (Mathonsi, 2011).The work by Chani (2008) concentrated more on finding a solution that most satisfies the service consumers based on the defined service level objectives. To ensure the most optimal service is selected a flexible Service Level Agreement (SLA) is used. This flexible Service Level Agreement was called the Consumer-initiated SLA (C-SLA). Using C-SLA the consumer requirements are considered before the SLA template in formulated. This provides more customer satisfaction compared to the provider initiated approaches. The work by Mathonsi (2011) addressed the issue of service selection. It argues that using only the service functionality for service selection is not sufficient but rather using it in combination with its QoS can almost guarantee the most optimal service is selected. A g-broker was introduced to assist consumers in service selection based on the requirements they have defined. The work also added support for providing QoS measurements to ensure no falsified QoS measurements are used in service selection. The main difference between the work by Mathonsi (2011) and the work reported here is that, in this work much emphasis is put on realising a collaborative environment for use by SMMEs in an attempt to providing assistance to develop new, innovative products and service offerings that meet the market needs and the other work assumes services are already available, all that is left is finding ways of using the most optimal services based on the measure of both the service functionality and the observed QoS measurements. Through building upon some previous work on GUISET, the work reported in this dissertation is the first effort aimed at demonstrating a GUISET configuration using an e-business scenario, where services must be composed at runtime applying dynamic binding to the services to be composed. The concentration was put on service composition and dynamic service binding to make sure service subscribers are only concerned with their business goal and not bothered with all the processes involved in achieving the business goal. Service consumers are only concerned with their interaction with the GUISET portal and do not really want to know how GUISET processes their requests. This work created a prototype using open-source tools to implement a typical application that explores how GUISET could support service composition and dynamic service binding. Our mode of service composition and dynamic service binding depicts an infrastructure that assembles components based on the desired business process and ensures that services used by the components are connected to on the fly. The idea is to ensure that all the services are needed are available in the GUISET repository of services.

2.3 GUISET Related Initiatives:

GUISET infrastructure is still in its evolutionary stage, although it is clearly defined to distinguish it from existing infrastructures. The main difference between GUISET and these other infrastructures are the contextual reference which has seen GUISET mainly focusing on providing an environment for the SMME to enable their IT presence. GUISET acts as a service market for SMMEs; these services are made available through a GUISET portal. A GUISET portal is the only point of access between its users, SMMEs and the services it offers. All the processing it carries out is hidden from the eyes of the user. The users access all the services that belong to the group they are members of or otherwise they can buy additional services from the service market.

Among the GUISET infrastructure related initiatives examples like; the IT Department (www.theitdepartment.co.za) and ASG (www.asg.co.za), can be mentioned. However, it is worth noting that no scientific evidence has been provided in these examples. This makes comparing them with GUISET difficult, as no validated report is provided. An autonomous home control system (Seinturier et al., 2012) and Akogrimo (http://www.mobilegrids.org/) on the other hand are other initiatives closely related to GUISET that provide evidence to research community. The IT Department is said to support small businesses by outsourcing cost effective IT platforms to them, providing support agreements and the installation of servers. Not much information is revealed as to how these platforms were implemented besides that Microsoft servers and networks were used in other services offered by the IT Department. It is not easy to determine whether these platforms can be as reliable as the GUISET.

The primary concern for ASG is offering IT support that is tailored to the needs of the business whilst ensuring data security is maintained. ASG tries to understand the goals of the business and in turn offer support that is closely related to the business objectives. Both ASG and the IT Department aim to support the same set of users as the GUISET initiative. The difference is that GUISET concentrates more to those that lack financial stability to subscribe in order to offer them access on a pay-per-use approach without having to own the

2.4 Motivating Scenario:

The challenges posed by service composition in dynamic and open environments are best explained through the use of an example scenario (Iyilade, 2010). A case study in the form of an e-commerce application supported by the GUISET infrastructure is introduced with the aim being on illustrating how a combination of tools is put together. The combination of tools applying service composition and dynamic service binding in computing environments can be demonstrated as the GUISET infrastructure. The scenario aims to give a description of a combination of components that are put together to define a business process and dynamically binding services that have been defined by the assembled components.

An initiative to assist the community of Nongoma with an IT infrastructure called GUISET aims to give its subscribed SMMEs a collaborative infrastructure where they can conduct their business activities online. GUISET provides the answer for NOS as one aspect of GUISET, which aims at giving SMMEs the look and feel of owning their own online stores where they can expose their products to their customers both local and abroad because of the capabilities the internet brings.

A Sample GUISET Application

Recently, Johnson, a resident of Empangeni, started a small business. The start-up capital has consumed all his investment outlay, he could neither afford to buy a R3 999 PC, nor pay the total cost of ownership (TCO), TCO for the PC. TCO is actually R10 000, being the price quoted by XYZ Computers in Richards Bay for Computer peripherals, office software and simple accounting tools. However, 10K is just the initial investment; the business must pay R1000 monthly, being the maintenance cost. While Johnson B&B will like to be able to have access to the Internet, advertise online, and accept online customer bookings, the business cannot absorb the overhead required to use e-Commerce and eventually transform into an e-

Business. The mortgage instalment on the building currently housing Johnson B&B is a commitment of another R1000.00.

GUISET provides the answer for this business because a Cooperative Group of arts and craft sellers have been formed in Longoma, 100km away, operating their micro businesses based on the GUISET infrastructure. All that will be required is for Johnson to apply for the membership of Longoma Cooperative Group. Basic membership subscription of R250 a month opens up the following opportunities for his business:

1. a web presence for Johnson B&B;
2. online booking for his customers;
3. an SMS notification for every reservation completed by a customer;
4. one monthly advert of special offers to promote his business; and
5. a basic accounting and reporting service which entitles the business to one end-of-month standard report;

This scenario outlines how the user interacts with the application to perform a transaction giving a clearer view of the activities taken to complete the process. Figure 2.3 depicts a typical infrastructure built from disparate components yet serving one business goal. The intention was to show that different levels of the system are used in the process. That is then, to complete the business process the portal, middleware and infrastructural levels coordinate, they cannot be used in isolation otherwise the system will fail.

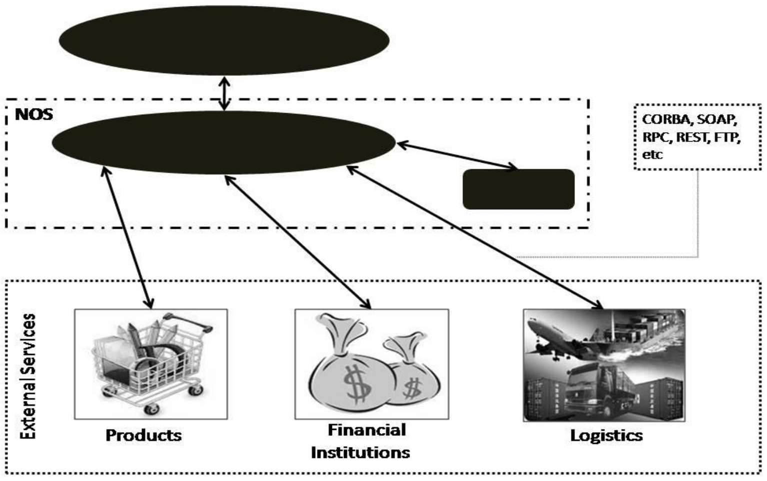


Figure 2.3: A Typical application using external services

The process taking place is deemed technical and is not visible to the eyes of the user. The only interest of the user is achieving his intended business goal and not knowing the inner details like the components that interacting to achieve it.

As illustrated in Figure 2.3, to process the customer's request, NOS web application will use both internal and external services collaboratively to complete the business process. To support the above scenario there is a need for an infrastructure that is built using tools that address service composition and enabling dynamic service binding. The usage of a modular approach gives an opportunity to deploying an infrastructure and its supported applications that are evolvable with time. Service composition is used for the assembly of components into a combination that interprets the flow of events completing a business process and dynamicity enables the coordination of these components to bind to services at runtime. To realise the GUISET infrastructure, much attention was paid to its characteristics, service composition and dynamic service binding. With the already mentioned characteristics, GUISET also aims to support software evolution. The software development approaches that have been selected for comparison in this work enable the application of a systematic approach when building software applications with possibilities of being evolvable over time.

2.5 Characteristics of the GUISET infrastructure:

The GUISET infrastructure is a multifunctional system that exhibits unique operational characteristics. In the preliminary research a few operational characteristics that distinguish the GUISET from other systems were identified:

- The underlying infrastructure configuration embodies a runtime environment in which components interact and their interaction is only limited to the information exchange;
- Resource coordination, meaning the infrastructure should apply some coordination technique to assemble business components conforming to some business goal.
- Dynamic runtime behaviour, meaning that the actual services used in components are not necessarily known until runtime, and the overall functionality of infrastructure emerges from assembling components;
- A dynamically-defined control, meaning that control over the system functionality is not necessarily owned by a particular component, rather, control changes based on which function the system is performing;
- An evolution of both the infrastructure and the application(s) running on it enables an infrastructure that is extended each time an additional functionality surfaces that it should support.

2.6 Chapter Summary:

This chapter concentrated on presenting GUISET as the foundation of this work, has started from a short introduction of grid and utility computing the core building blocks, then to the architecture. This has been followed by the research issues that must be addressed in the context of GUISET. A motivating scenario for an e-business application has been presented as a foundation of what an infrastructure should support to be regarded as a GUISET infrastructure. Lastly, the characteristics of the GUISET infrastructure have been presented to build the foundation for the formulation of the GUISET-based evaluation framework.

Chapter 3: Software Development Paradigms

3.1 Introduction:

This chapter introduces a number of possible software development paradigms that can be used as a systematic process for the development of the GUISET infrastructure. Since the inception of the software engineering field, many software development paradigms have evolved to be applied in one project or the other. Among the many available software development paradigms available, this study only focused on the following paradigms: Aspect Oriented Programming (AOP), Component Based Software Engineering (CBSE), Service Oriented Computing (SOC) and Resource Oriented Computing (ROC) as possible candidates for realising the infrastructure. There was no criterion put in place for coming up with the candidate paradigms. It was mainly based on the availability of supporting materials such as research publications and documentation.

This chapter begins by presenting the requirements an infrastructure must successfully address to be regarded as a GUISET infrastructure in Section 3.2. This is then followed by an overview of different paradigms shortlisted for the development of the GUISET infrastructure in Section 3.3, which is then followed by an analysis that leads to the selection of the methodology used in this work in Section 3.4. The selection is made possible by using the GUISET-based framework. Lastly, in Section 3.5 the chapter presents the current state on software development paradigms.

3.2 GUISET based Comparison Framework:

In this section, a brief description of an evaluation framework based on the GUISET requirements that should be met for the software development paradigms to realise the infrastructure is presented. Based on the GUISET infrastructural requirements a comparison is made among these software development paradigms: AOP, CBSE, SOC and ROC to find one paradigm that is closest to addressing the requirements.

The comparisons have been informed by general literature and its applicability in some reported work or tool observation. General literature in this work refers to knowledge that has been gathered from materials concentrating on laying the foundation on the paradigm. The GUISET based framework was formulated to realise an infrastructure that supports both service composition and dynamicity. For the framework to aid in the realisation of the GUISET infrastructure it needs to address the following criteria: (1) modularisation, (2) assembly of business services, (3) technology agnostic service access, (4) flexible and dynamic environment, and (5) changes based on the availability of business services.

1. **Modularisation of business services**: the methodology should provide support for building individual modules. Hence the infrastructure can be built from scratch by simply integrating different modules providing some functionality.
2. **The assembly of business services**: the methodology should enable support for the assembly of business services to complete a business process of the users' consumer. Hence the infrastructure should be inter-operable to support configuration of services into a composite to complete the process. In this instance business services are not necessarily services, but rather resources that meet the business goal.
3. **Technology agnostic service access**: the heterogeneity of protocols is diverse; to support services implemented using various languages and technologies, there is a need for accommodating a multiprotocol service discovery and access for the different communication paradigms. Hence there is a need for the infrastructure to support multiple protocols for the discovery and accessing of services.
4. **A flexible and dynamic environment**: customisation and functionality of services at runtime is essential for an application to adapt to changes. Hence configuration and dynamic adaptation are essential functionalities in the infrastructure.

5. **The availability of tool support**: As the user selects a particular product, several business services might be available to support such a transaction. Hence the infrastructure must be able to support changes in the services and reduce the impact of the changes in the user activities.

The software development paradigm closest to addressing the requirements stands out to be a suitable candidate for the implementation of an infrastructure deemed to be a GUISET infrastructure. To give a fair and clear justification of the paradigms, some foundational background on the candidate paradigms is presented in the following section. The aim is to remind or build on understanding of each paradigm before proceeding to the comparisons.

3.3 Software Development Paradigms:

Software development paradigms have been around for a very long time and have always been the main focus in the software development lifecycle. A methodology is generally a guideline for solving a problem using specific components in a given discipline. The evolution of new paradigms is usually caused by the existence of problems that have been perceived not to be solved by existing paradigms or to just broaden the research findings in the field of software engineering. When it comes to the selection of a development paradigm there is no best paradigm among all, but they can be best classified in relation to the project they are applied in. Hesari *et al* (2010) states that the influx of a variety of software processes and software development methodologies has made it difficult to select a methodology for a specific project or to construct the appropriate methodology through chunks of assembled methods, and that is why the evaluation of paradigms has become an essential task. Dahiya (2010) stated that when software development was first initiated it was not as structured and well documented as it is today.

The selection of the paradigms that affords this comparison is based on the availability of documentation that describes it and time span that the methodology has been developed.

Even though documentation does not guarantee software building success, it does make it easier for developers to quickly understand the use and support of the methodology. It also affords the developers the opportunity of defining documentation for systems they are building to be easily understood and possibly support changes with ease. This documentation was mainly introduced to try and avoid any software development failure which is perceived to be the main cause, if no methodology had been applied. The documentation lays the foundation for the approach taken when using the methodology even in different projects with ease. Effective usage of a methodology is achieved by following its defined processes in an accurate and consistent manner across all projects.

This section intends to build an understanding of each of the development paradigms that has been reviewed to make the selection by presenting its foundational background. The aim is on establishing enough knowledge on each of the paradigms and how they are applied.

3.3.1 Aspect Oriented Programming (AOP):

Aspect-oriented Programming (AOP) compensates the shortcomings experienced with using Object-Oriented Programming (OOP). An OOP application program is made up of a series of objects which are the basic modules of the application communicating together through the exchange of messages. The basic techniques for building OOP applications are inheritance and polymorphism. These are important attributes of an OOP application as they enable building of applications that exhibit reusability and extensibility capabilities. Zhengyan (2011) states the problem with the OOP paradigm is trying to import some common actions into objects that do not necessarily need the inheritance relationship with each other. This leads to overcrowding of code and limits re-usability of resources.

AOP is a programming paradigm which deals directly with aspects of concern rather than modules of software code. AOP separates its code in terms of the code that deals with the core business logic from that which spans across objects. Kiczales and Mezini (2005) state modules are available; all that is needed is for the composition of interfaces to be established that uses these available modules. In fact various techniques have emerged with AOP as required, namely that AOP is a mechanism proposed to enable modular implementation of crosscutting concerns. It, AOP, addresses the management of functionalities in the system by encapsulating dispersed functionality into well-defined modules in the configuration of a system. Separation of concerns is a well-established principle in SE. A concern is a part of a problem that is treated as a single conceptual unit. Java provides perfect support for AOP that has led to AOP being supported and applied on different platforms. AspectJ was developed using Java to conveniently develop AOP software (Sirbi and Kulkarni, 2010; Zhengyan, 2011). AspectJ is based on both AOP and CBSE with each methodology applying its principles where concerned such as CBSE deals with modularisation and AOP deals with the separation of concerns.

AOP forms the basis for Aspect-Oriented Software Development (AOSD). AOSD was proposed as a technique aiming at improving the separation of concern when designing systems and support for an improved evolution and reusability. Modules make it possible to make decisions on the use of the module by only looking at its implementation and its interface without worrying about the rest of the modules it will be used with. Deiters (2005) reports on the attention that AOP received from the Microsoft .Net development community where it shows how AOP has been used in web services illustrating some common problems that AOP can address.

3.3.2 Component Based Software Engineering (CBSE):

Component-based development (CBD) enables software systems to be built through the composing or assembling of software components. CBSE is an established approach in building software systems in many domains such as distributed systems; web based systems, desktop applications and embedded systems amongst others (Breivold and Larsson, 2007;

Masek *et al*, 2009). Component Based Software Engineering (CBSE) aims at accelerating software development and promoting software reusability and maintenance through the assembly of business components to meet certain business goals. Components are the building blocks of building component-based software systems. Many views have been provided on what a component is perceived to be, the common consensus regards a component as a black-box entity with defined interfaces and behaviour that can be reused in varying contexts and with no prior knowledge of its internal structure. One of the benefits of using CBSE as development process is the ability to re-use readily available software components in the development process which means only developing components that are specifically exposing functionality of the intended project, if not all components are readily available.

Although CBSE has worked in deploying re-usable components, it falls short of providing mechanisms for working with readily available software components. The software components used in the CBSE development are built conforming to an architectural model. Much information is known of the service before it can be used, in essence it can be said CBSE is a technique for modelling and assembling a specific piece of software. In a small scale application using readily available services can be optimal, but as the application grows more it becomes difficult as more and more services are needed, varying protocols and the various devices used. Service orientation handles these issues not addressed by CBSE through the use of services made available in the service market. SOA uses service exactly as they are regardless of their service implementation and technology used, allowing the usage of web services and legacy applications.

Although component-based software engineering addresses some requirements of service-orientation, it still does not address its key elements, such as accessing services geographically dispersed and assembling them according to some business process

(Papazoglou *et al,* 2006). This is a drawback as the software development process is quickly moving the direction of using functionality that is provided through services located somewhere else.

3.3.3 Service Oriented Computing (SOC):

Service Oriented Computing (SOC) is a SE paradigm aiming to support the development of rapid, low-cost, and easy composition of distributed applications even in heterogeneous environments using services as its key abstraction though service composition (Papazoglou *et al*, 2006, Ramollari *et al*, 2007; Iyilade, 2009). SOC is based on SOA which is an architectural style for building software applications that use available services on the web (Iyilade, 2009). Service orientation is currently one of the most appraised paradigms in SE for building enterprise applications (Koskela *et al*, 2007). The service orientation paradigm aims at performing business processes ranging from simple to very complex interactions between components to complete the process. The activities performed to achieve an SOC infrastructure helps the organisation develop meaningful services, service compositions and techniques for managing these services.

Current research in SOC among others has focused on service foundations, service compositions and service management and monitoring (Papazoglou *et al*, 2006). SOC is an evolution of distributed computing which has been designed to achieve an interaction between software components using services across the network. These services are assembled together to create new applications. By using the service composition mechanism the assembly of services follows some business logic to achieve the business goal of the newly created applications. Most importantly, using SOA, a single service may be shared across a number of different applications achieving greater reuse of existing services. In general, the application of SOA can be used internally and between Information Technology (IT) network equipment, services infrastructures across functional domains and geographies for the extended enterprise, including B2B, with the maximum re-use of existing technologies where it makes sense (Papazoglou *et al*, 2006; Kryvinska *et al*, 2010). Davis (2009) states a service just like a component is a self-contained unit of functionality.

Blinco *et al* (2009) and Xiong-Yi show that SOA can be defined in different ways depending on the perspectives of the business owners. These definitions can take the form of business aspects, technological aspects, and from the view of the IT management. These perspectives show that the success of SOA is perceived differently depending on who views it. Papazoglou *et al* (2006) stated that SOA is a logical way to designing software systems to provide services to either end-user applications or to other services distributed across the network. SOA uses web services as its building block with the aim of enabling loose-coupling; this in turn can make services reusable. SOA is regarded as the substitute of the traditional architectures that are tightly-coupled, technology agnostic and object-oriented; it enables heterogeneous systems to share business logic and information with its business partners outside of their business domain (Xiong-Yi, 2009). Managing loosely coupled services in SOA is the strongest requirement. It enables services to be built independently of its users. Web services support the new generation of e-business application, as it can enable the building of applications that can communicate within themselves (Najdawi, 2009). Open issues regarding service foundations, reports the need for an infrastructure supporting diverse service messaging models, consistent with SOA interfaces with the capabilities for transmitting and performing transactions necessary on the given information (Papazoglou *et al*, 2006).

SOA is one of the most popular topics, which is an architecture building method used to describe, link and integrate reusable business services with clear boundaries and self-contained functions. Functions are described as well-defined services that can be used to compose working applications.

Although there are many definitions coming with the acronym SOA, the World Wide Web (W3C) refers to it as “A set of components which can be invoked, and whose interface descriptions can be published and discovered”.

One of the most important questions raised when implementing SOA, is related to the orchestration concept. Orchestration involves a step by step approach for the invocation of several services and combining them. The orchestration may be understood in more than one way. Firstly orchestration is related to Business Process Management (BPM): how can application logic be modelled and built for business applications requiring the invocation of several services in their core business functionality and secondly the meaning of orchestration is related to service composition. Service composition is an aggregate of services assembled together to perform a particular task or business process. This is achieved through some mechanism that has been described in the SOC paradigm. Maigre (2010) states that since the introduction of web services research has focused on the use of services: creating service descriptions, finding services from a pool of services and building as well as executing compound services.

3.3.4 Resource Oriented Computing (ROC):

Resource Oriented Computing is a fundamental model for describing, designing and implementing software systems through the usage of resources. In the context of ROC a resource is defined as a set of information. Designing and implementing a large scale evolvable enterprise software system is a challenging task. Component-based and service-oriented development techniques have become the key mechanisms for building such a software system in a timely and affordable manner. Component based software engineering and service oriented computing as the two most dominant engineering paradigms in the current software community and industry are similar in some way in the approaches and techniques they use in building software systems, which has led to some confusion in understanding and applying the concepts in a correct way (Breivold and Larsson, 2007).

Resource Oriented Architecture (ROA) is an architectural style which focuses on using resources as its building block. ROA is typically implemented using RESTful services. The essence of Representational State Transfer (REST) as an architectural style for developing distributed systems such as the Web is to focus on creating a services environment that is loosely coupled to enable reusability of services to be maintained (Guinard *et al*, 2010). REST is the core of the Web and uses URIs for identifying and encapsulating services available on the Web. In ROA the request-response implements interaction is defined around the transfer of "representative" resources (Hu and Shan, 2010). In ROA, a resource interface is a prerequisite for enabling access and state manipulation of a resource. The HTTP operations are normally used for managing resources. These operations are PUT: which is used to update the state or creating a resource, GET: used to retrieve the representation of a resource, POST: which creates a new resource, and DELETE: used to remove a resource. Overdick (2007) states that even though HTTP is dominant in ROA there is no strict adherence that must be maintained with HTTP applications with many applications violating the concepts of resource orientation.

3.4 Comparison of Software Development Paradigms:

Specifically the aim is deducing which development technique and its supporting techniques are suited in putting together an e-enabling infrastructure, be it an e-commerce, e-tourism, e-booking, etc. using available technologies. To make a justification for the selection of the paradigm and its supporting tool, literature was consulted to get a clear understanding of how it addresses some of the requirements, which are the GUISET-based evaluation criteria.

GUISET, as one of the distributed applications in this scenario, is structured as application components and services interacting with each other. Business services either local or remote are used in the assembled components for achieving the business goal of each composite. There is a need to give some clarity between “services” and “components” to avoid some ambiguities in understanding these concepts especially as being addressed in this work. The terms services and components are widely used in the field of software engineering, which means we need to make a clear distinction among them in terms of the aspects they seem to capture. Although these concepts seem very much similar the favoured difference is that described by Bocchi *et al,* (2008), which takes a view from CBD, of a service as a way of orchestrating interactions among a subset of components in order to obtain some required functionality, which means the coordinated or assembled components use services to accomplish their specific tasks.

For an infrastructure to be qualified as a GUISET infrastructure it must meet the characteristics as defined in chapter one. The following discussion highlights how each paradigm addresses the requirements of the applications’ supported by the GUISET infrastructure.

1. **Modularisation of business services**: All the candidate paradigms are in support of modularisation; the difference is they have shown somewhat different concentration levels in terms of addressing it. Papazoglou and Georgakopoulos (2003) and Bocchi *et al* (2008) stated SOA enables a flexible interconnection of autonomously developed and operated applications discovered according to the required levels of service. Both CBSE and SOC enable full support as they strive for the re-using of available resources. The difference between the two is CBSE modules are tightly coupled to the environment with less or no reusability while SOC modules are loosely coupled with high re-usability. In AOP business logic is broken down into distinct parts and these parts are called crosscutting concerns. Crosscutting concerns are meant to address some functionality with ease and manageability in software applications. AOP

achieves modularisation by adding an extra abstraction mechanism called Aspects on top of the existing modularisation mechanism. Aspects can tackle the problems of scattering and tangling of code by reducing the spread of code belonging to a certain concern over different components.

To best achieve modularisation in other implementations it has been reported AOP is combined with CBSE with the former addressing separation of crosscutting concerns and CBSE addressing the componentisation part of things. Work such as that of Eichberg (2005) shows how AOP has been combined with CBSE to provide both the principles of crosscutting concerns and modularisation. With ROC resources are available as logical units, but not much information is provided as to how modularisation is supported. The idea behind ROC is on providing resources when needed.

2. **The assembly of business services**: Throughout the surveyed literature on the AOP paradigm no report on the assembly of services was encountered with ROC showing minimal support for the assembly. On the other hand, both CBSE and SOC provide full support, components need to be designed in a manner facilitating the subsequent assembly process when components are assembled together to develop software systems. In CBSE, the selection of components that deliver services addressing some business goal is a design time activity (Bocchi *et al*, 2008) compared to SOC where it is done at both design and runtime (Iyilade, 2010), depending on where and how it is being applied. The only report depicted in this work where AOP is involved in the assembly of services, is when it is used in combination with CBSE with AOP addressing the separation of concerns in the assembly created using Component Based Software Engineering. Beyond that, no other work could substantiate whether AOP does support the assembly of business services. ROC's concerns go as far as

providing the necessary resources; it does not worry much about how those resources are going to be used whether in an assembly of other resources or as a single entity. In ROC it is strongly acknowledged that in the process of service selection, services up for selection might not have the necessary information to make the comparison necessary to select one service over the other. ROC comes to the rescue by making every entity explicit not just as services. Such an explicit entity is then called a resource.

3. **Technology agnostic service access**: Only SOC reports full support for heterogeneous protocols, which makes it easier to use services implemented in any technology without tampering much with it, whereas with CBSE the component interaction is restricted to those implemented in similar technologies. The primary concern of the remaining paradigms ROC, CBSE and AOP only concentrates on providing the necessary functionality and they are mostly used in tightly coupled environments. Overdick (2007) stated the evolution of ROA was due to the fact that not much details are always given of the service, so ROA solved this dilemma by making every entity explicit not only as just services and they use a universal interface to establish communication. ROC with its capability of recognising "services", "objects" and "data" as resources makes it technology agnostic. CBSE does address issues pertaining to the nature of technologies used in the implementation, it is mainly formulated through components and services implemented in the same technology.
4. **A flexible and dynamic environment**: Both ROC and SOC are flexible enough to provide a replacement for non-existent resources which previously existed. Cheesman and Ntilozalos (2004) and Papazoglou *et al* (2006) discuss flexibility as one of the core characteristics of an SOA application with its ability to support the

reconfiguration and updating of a business process based on the new requirements the business process should address. This flexibility is achieved especially in loosely coupled environments where tampering with one component does not necessarily lead to more components being tampered with to meet the whole business goal which both AOP and CBSE do not address. In CBSE, the resources in use are known at design time as they are tightly coupled to the environment, this causes failure to the system should they not be available when needed. Flexibility of this nature is outside of the scope of AOP as there is nothing reported on it and understandably so the issues addressed by AOP are somewhat different.

5. **The availability of tool support**: the support for loose coupling of services starts at such a low level as a development paradigm that has been applied supporting it. Both SOC and ROC have extensive support for providing an environment where services are accessed only when needed due to its loosely coupled nature whereas with CBSE tight coupling is applied. One can almost argue the tools supporting applications developed on the principles of SOC can also support ROC. This comes from the knowledge that the major difference between ROC and SOC is that the former treats everything as a resource and the latter treats it as a service. Work by Guo *et al* (2010) shows the similarities in the software development using ROA and SOA approaches. They both use Web services as their implementation strategy the only difference being in ROA where these Web services are wrapped around resources. The extent of tool support in CBSE and AOP is sufficient with its tight coupling nature of development. There is need for also supporting tools in any development. In light of this it becomes difficult to conclude both CBSE and AOP address tool support as the reported work shows the tool support is only relevant to tools developed to validate

some phenomena rather than having tools readily available to be utilised based on the concept of the paradigm.

The table below summarises how each paradigm is supported based on the framework. The table is formulated based on the arguments from the comparisons made and some weights were given based on the perceived support each paradigm offers or does not offer as per requirement. The legends are:

“++”: excellent “+”: good “0”: satisfactory “-“: poor

Table 3.1: Software Development paradigms comparisons

	Modularisation	**Assembly**	**Agnostic**	**Flexibility**	**Availability**
AOP	++	0	0	-	-
CBSE	++	++	0	0	+
SOC	++	++	++	++	++
ROC	++	+	+	++	+

Based on the reported work and observations of some tools applying some paradigm SOC has been found to provide full support. CBSE and ROC are the next closest in satisfying all the requirements with the SOC just edging them over as already mentioned.

The table summary above shows SOC as the most optimal software development paradigm to use in deploying the GUISET infrastructure, it exhibited more support than the rest in the comparisons. SOC is a paradigm or model which many enterprises have exploited for many years as a source of delivering business applications. Interestingly enough, SOA is not a completely new approach but rather its firm foundations are built from CBD and Design by Contract (BbC). DbC is an approach to software development based on defining formal, precise and verifiable interfaces for software components. The only problem is they just fall short of fully supporting SOA which is mainly based on addressing issues of loose coupling, runtime discovery, and technology independence. In the realisation of SOA based applications the understanding of the business-application-technology stack plays a vital role (Figure 3.1). The figure aims to depict the relationship among the different layers with the Business Layer automated by the Application Layer which applies the Technology Layer.

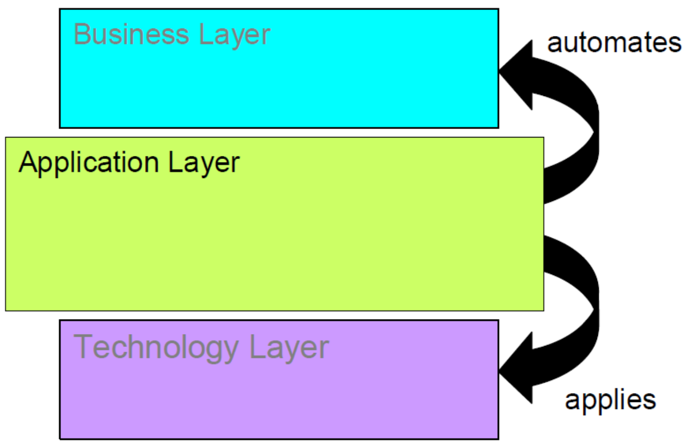


Figure 3.1: The Application Layer - bridging business and technology [Cheesman and Ntinolazos (2004)]

Software applications are the bridge between business decisions and the technology used. The existence of software applications aims at automating some part of or all the business processes with the intention of realising more efficient and effective processes. The Application Layer has somewhat the most difficult job of acting as a bridge between the two layers, Business Layer and Technology Layer as they are continuously changing and at all times, it should address those changes. For the Application layer to successfully work with the other layers it should possess the following characteristics:

- Flexibility: the ability to update or reconfigure existing business processes to support additional functionalities as required by the business processes.
- Technology-independence: business process implementations should avoid depending on some implementation technology. This should include the ability to use legacy

systems, COTS and new services as well as accommodating multiple technology platforms.

There have been several attempts to make the transition to SOA an enjoyable experience with a huge variety of tool support and some programming models. In SOC a programming model is a collection of abstractions, models, techniques and supporting tools with the aim of assisting developers of SOA applications. These programming models simplify the modelling, assembly and deployment of SOA applications. A programming model enables even developers with minimal programming skills by introducing well-defined component types that model common kinds of artefacts that developers produce and deploy into solutions. To mention just two programming models applying the SOC principles as reported by Nigul *et al,* (2009) there is Service Component Architecture (SCA) and JBI (Java Business Integration).

This work is not the first in making comparisons of available paradigms, but rather this work concentrates on making the comparison informed by the infrastructure requirements where it is applied. The next section presents some of the work where comparisons are the main objectives of the work and possibly bring some light towards the evolution of these paradigms, the transition from one paradigm to the other if there is some relationship towards the different kinds of these paradigms.

3.5 Software Development Paradigms: State of the Art

The status quo of the software development paradigms gives an indication that the evolution of these paradigms is mainly based on addressing issues previously not addressed by existing ones or completely going another route all together. Since the inception of software development paradigms more and more developments are now based on trying to achieve as much re-usability as possible rather than developing from new application from scratch.

There are relatively few complete experimental results comparing the effectiveness of software development paradigms. This makes it hard to select a suitable paradigm for the right project. This work assumes the effectiveness of each methodology can be found by studying how it has been applied in the reported. The application of a methodology aims at improving the end product of the development process by developing better information systems (Avison and Fitzgerald, 2006). Solving ad hoc problems has also led to some paradigms being combined to realise the software product. Combining paradigms has somewhat created some confusion when it comes to determining what exactly each methodology aims to achieve and not. So as to make the evaluation fair, where paradigms are used in a combination, it is clearly identified what aspects each methodology covers. Works such as that of Eichberg (2005) is a good example, where Alice (Eichberg, 2005) is introduced applying the principles of both CBSE and AOP with the latter concerned with separation of concerns and the earlier deals with modularisation. This ensures each methodology stands out in terms of the features it provided in the development process.

The current state of the development paradigms as presented in this section sees SOC gaining more ground in terms of its use in deployment of applications compared to other paradigms. Software development paradigms were introduced as a response to tackling the high complexities arising with the development process of software products. These are just some of the few works analysed during the process of establishing the framework for this work.

Guo *et al,* (2010) discuss the opportunities that can be achieved from developing a software application based on SOA and ROC. The design of the application is on SOA but also follows ROA principles. Since ROA uses resources for the exchange of data, in this work the application architecture is built in which the resources and their operations are packaged to Web services. The implementation strategy applied which is based on Web services is deemed to be that of SOA. Also ROA resources can use it through a defined wrapping.

Besides the usage of Web services for SOA and resources for ROA the development strategy is the same. Data and its operations are published as Web services and can be shared among the enterprise software application fully and safely. The idea behind using SOA and ROA combined has to do with gaining flexibility of using any resource either than the default support of web services by SOA.

Hesari *et al* (2010) introduce an evaluation framework as an approach to deriving the contributions made by a methodology. The framework can also act as a foundation for comparing paradigms to make an optimal selection based on the desired goals. The desired goals always differ based on the stakeholders request even for software applications performing the same functionality. The evaluation establishes a better understanding of the features, strengths and even weaknesses of the methodology. This understanding can influence the selection of a methodology as much detail would have been gained based on what to expect on each methodology given the project requirements it should fulfil.

Palacio (2010) in the work aiming to understand and presenting the development process of a Data-driven Support Systems (DDSS) in an organisation adopted a framework for comparing paradigms introduced by Avison and Fizgerald (2006) of which by its authors is said to give a set of features that prove to be a reasonable guide in comparing paradigms for selection. This framework applies seven elements to get a comprehensive description of the methodology being analysed, from a set of principles that underlie the methodology to tool support for specific techniques or complete paradigms.

3.7Chapter Summary:

This chapter started by presenting a framework for comparison, then the candidate software development paradigms that can be used to craft the GUISET infrastructure. Then, SCA as a chosen programming model has also been presented. The selection of a methodology to be used was made possible through the GUISET based requirements and the open source solutions used.

Chapter 4: Developing the GUISET infrastructure from Existing Open-Source Solutions

4.1 Introduction

This chapter introduces the formulated conceptual architecture which intends to support service composition and dynamic service binding. This architecture is based on the GUISET characterisation of Section 2.4 and from the outcome of the selected software development paradigm in Chapter 3. This is done by first outlining the design requirements that informed the formulation of the architecture to support service composition and dynamic service. Then, the concepts of software development and evolution are also presented as a mechanism for the realisation of the GUISET infrastructure with the intentions of having an infrastructure evolvable over time to accept additional functionalities with minimal hassles.

The chapter begins by presenting the GUISET Service Composition and Dynamic Binding Architecture outlining the building blocks and what the infrastructure intends to support in Section 4.2. This is then followed by the software development and evolution concept and open source solutions to indicate the approaches used in the context of this work in Section 4.3 and 4.4 respectively. Finally, Section 4.5 introduces the current state of concepts discussed in this chapter.

4.2 GUISET Service Composition and Dynamic Binding Architecture (GuSCaDA)

The GUISET infrastructure is envisioned to be an environment supporting collaboration of service components and dynamically binding services defined in the collaboration to realise the desired goals of the infrastructure to be supported. The GUISET architecture as proposed by Adigun *et al* (2006) envisions an infrastructure formulated through a combination of services components to address some business goal. The GUISET is based on a concept there is an affordable technology. What is needed is an appropriate strategy to make these technologies available (Adigun *et al,* 2006). The middleware layer as envisioned assembles service components and in turn dynamically binds services defined in the assembly at runtime to complete the business process. GUISET follows the modular approach in the development to promote an evolvable infrastructure where additional modules can be added to provide additional functionality without disrupting the whole infrastructure. This research is aimed at prototyping the GUISET architecture using available technologies. Based on the research question outlined in Chapter One, Section 1.5, there is a need to create a GuSCaDA architecture showing the interaction of components in development of the GUISET infrastructure. This chapter, in Section 4.2.1, Section 4.2.2 and Section 4.2.3, introduces the GuSCaDA design requirements, the GuSCaDA architecture and the GuSCaDA operational and functional respectively.

4.2.1 GuSCaDA Architecture design requirements

The GUISET motivating scenario of Section 2.3 and together with the GUISET characteristics in Section 2.4 informed the design requirements leading to the formulation of GuSCaDA. The GUISET infrastructure as ICT enabler considers the following design requirements:

i. Composition of services

   In order to support applications running on the infrastructure, the service components need to interact with one another in a coordinated way to support the desired business process. To enable a coordinated interaction of service components, an assembly of components needs to be formulated showing a flow of components interacting in the achievement of the business process.

ii. Dynamic service binding

   The assembled components define the interfaces of services that need invocation at execution time to complete the business process. For a flexible management of service, registration and discovery services need to be stored in some registry.

GUISET intends to manage a number of services providing some functionality. With that being the case, there is a need for a mechanism that dynamically binds services that have been defined in the assembly. The approach taken by this work is binding services of any communication protocol without having to alter any configurations.

iii. Evolution of both the infrastructure and its supported applications

Both the infrastructure and the applications running on it should be subject to further developments without disrupting much of it configuring new functionality. The modular approach is applied in the development to ensure there are no dependencies among the interacting components and also affording reusability components.

4.2.2 Design of the Architecture

As already stated, the intention was to prototype a GUISET infrastructure supporting (i) service composition; and (ii) dynamic service binding of the composed components in (i). The infrastructure and its supported applications are evolvable to cater for additional functionalities without having to develop a new infrastructure from scratch which is now a common practice in software engineering and this work also builds upon that. The optimal solution for achieving such an evolvable environment is the application of a modular approach in the development with each module satisfying some aspect of the infrastructure and can be added or removed without disrupting the whole infrastructure configuration. GuSCaDa follows a modular design approach which is composed of these four major components the (i) *Assembly Manager* component; (ii) *Service Binding* component; (iii) *Dynamic Runtime Manager* component and (iv) *Service Registry* component (see Figure 4.1). The role of each of the components is discussed as follows.

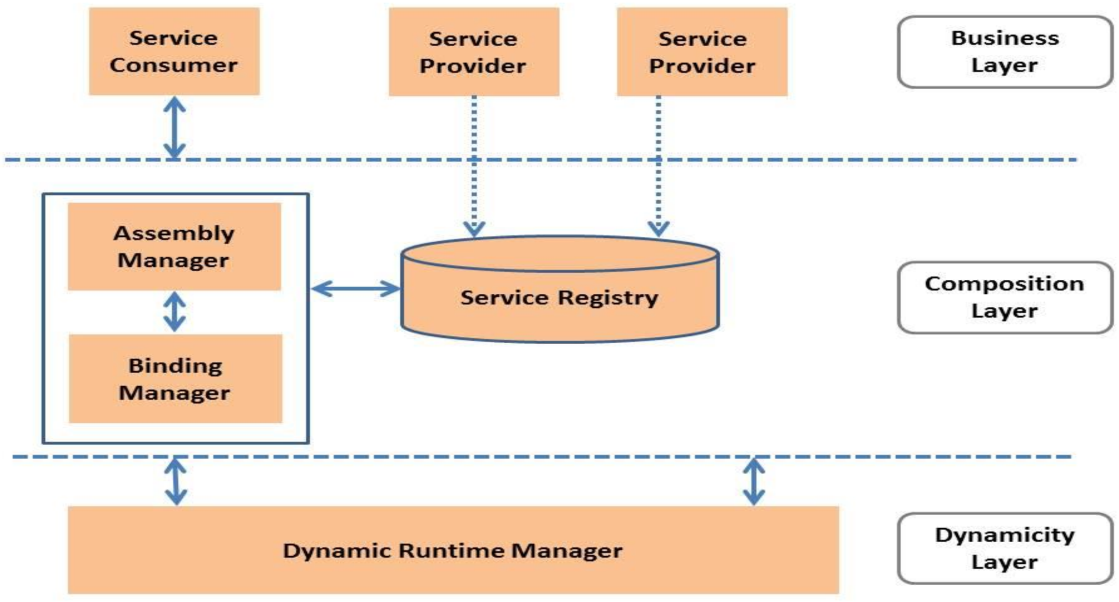


Figure 4.1. GUISET Services Composition and Dynamic Binding Architecture (GuSCaDA):

The Assembly Manager is responsible for assembling business components based on the desired business process by i) extracting the business logic of the process, ii) defining service components used by the process through well-defined interfaces, iii) creating composites based on the process to achieve its business goal.

The Service Registry is a repository for the storage of service descriptions registered by the service providers. The assumption is all the services needed for providing the necessary functionality are all available.

The Service Consumer can either be an application or another service that intends using services in the Service Registry for the purposes of executing, completing and achieving a particular task. A combination of services is used to formulate the process needed to achieve some intended business goal of the infrastructure.

The Service Provider registers its services in the service registry along with all other information about the service that is used for consideration in selecting the service.

Upon successful creation of a business process, the Binding Manager ensures the control of over the different services' technology implementations to ensure any service can be used by i) inspecting the defined composite to gather the necessary specifications of the services, ii) providing support for multiple protocols for flexibility among the choice of service implementations providing the required functionality; and iii) dynamically binding services

Once the Assembly manager has created a composite defining some business process and the Service binding has handled the multiple protocols presented with the service implementations, Dynamicity manager ensures services are discovered at runtime by i) using local services or ii) accessing remote services.

4.2.3 Operational and Functional Description:

The following description is used to walkthrough a typical application for service composition and dynamic service binding. There is also a clarification as to why service composition and dynamic binding of services are treated separately. We will assume a client of the GUISET infrastructure is an SMME needing an assembly of service components to meet its business goal as of the scenario of Section 2.3 and the client of an SMME uses the application supported by GUISET, the infrastructure. The SMME client specifically wants to buy products online and the GUISET client wants to offer its products online. To support both clients, GUISET client, an SMME exposes its business with the GUISET infrastructure which assembles service components based on the desired business process of the SMME and the SMME client, the customer, buys the intended products from an online store which executes the business process of the SMME in the GUISET infrastructure already defined. It dynamically binds services to the defined composition of service components.

An itemised list of the major interactions between the system actors and system itself during the creation of the process is described as follows.

1) The GUISET client registers its business activities with GUISET by outlining the intended business activity to be achieved;
2) The GUISET system administrator establishes an assembly of service components based on the process needed to complete a transaction defined in 1. It mainly concentrates on assembling the business logic by defining the interfaces and how they interact. This in turn gives the SMME the feel of owning an online store;
3) A customer buying products online through a store supported by the GUISET infrastructure triggers the registered business process of an SMME which is already assembled, in the process of executing the transaction the assembly dynamically binds services defined by the interfaces to complete the transaction;
4) Assuming the previous step has been satisfied, the SMME is billed on the services used by the assembly in processing the transaction.

To realise GuSCaDA, prototyping seemed as the viable approach as the intention was deploying an infrastructure that will undergo a series of iterations until the final product is achieved.

In the previous chapter, when the comparisons had been completed it was stated SOC was the closest in addressing the requirements to be met for an infrastructure to be regarded as a GUISET infrastructure. SOC as a concept is very broad and sometimes its application in development can be a daunting task. One of the ways of trying to simplify the application of SOA in development is through the usage of a programming model. As already mentioned earlier it provides a systematic way of developing applications. SOC offers some programming models such as SCA and Java Business Integration to try and simplify the development process. SCA and Open Service Gateway initiative (OSGi), the GUISET infrastructure solutions, are also presented as subheadings of Software Development and Evolution.

SCA and JBI are SOC development standards offering both commercial and open-source tools in their implementation. The JBI specification as developed under the Java Community Process (JCP) is an approach to developing SOA application. The JBI is built on the Web Services model providing a pluggable architecture for a container. The main benefit of using SCA over the other approaches is it provides a technology-agnostic programming model that decouples the components implementation from their communication protocols, allowing reuse at a higher level. Software applications developed following the principles of SCA should be deployable in different SCA vendor platforms, integration rules and deployment patterns without making any changes to it.

The architecture presented in Section 2.6 and the paradigm selection in Section 3.5 gives way for the selection of the open-source solutions presented here enabling the building of an infrastructure and its supported applications that are evolvable in due time. The GUISET infrastructure should accept additional modules to extend its functionality without tampering with whole infrastructure, which should be the same for the applications running on the infrastructure. In this day and age the development of software applications strives for building from existing components rather than having to create applications from scratch (Al-Jaroodi *et al*, 2010).

4.3 Software Development and Evolution

Software engineering research over the years has traditionally focused on methods, concepts and techniques generally applicable. Extensive research has been conducted in the field of software engineering, among others; the focus here is in the area of software development research. Software development is simply defined as a set of activities taken to build a software application. Software applications are developed to serve diverse purposes. Among the various reasons a software application can be developed, the most common ones are for personal use or to meet the needs of some potential users. Software applications have become a vital component for a number of business enterprises as a means of strengthening their competitive advantage by exposing their business in the e-business arena. The tasks that have been done manually before are now computerised gaining efficiency and accurate data being transmitted. Typically, these software applications are built to automate business activities. In turn less time is consumed compared to using manual labour together with reduced costs especially that of labour. To minimise the errors in software development, there are processes that have been introduced to act as guidance throughout the entire project. According to Sommerville (2001) the generic activities in software processes that must be followed are:

- **Specification**: defines the functionality of the system
- **Development**: translates the specification into a system
- **Validation**: ensures the system is as expected
- **Evolution**: continuous changes with the changing needs

Most often adhering to these processes can lead to a software production success. It should be noted that the selection and use of each process depends on the software application that must be produced. Nowadays software applications being produced must evolve over time. It should be able to adapt to changes of the user needs by allowing further developments. It should also be able to integrate or find a mechanism of using any available legacy systems with ease should the need arise. This makes software development one of the most complicated tasks performed by humans with the growing complexity of software applications being developed. This has led to a lot of standards and frameworks being created to assist in decreasing the amount required in the development process and combating the complications that arises with complexity. Various parameters including costs, training time, and future supports are the main considerations for companies to select development tools and frameworks. The approach taken by this work entails applying an approach to the development that the will enable the software application to further evolve with additional functionalities being introduced. This section discusses the development approaches that have been applied in the development of a GUISET infrastructure that is evolvable over time namely SCA in Section 4.3.1 handling the assembly of business components at design time and OSGi in Section 4.3.2 addressing the issue of dynamically binding to the services defined by the assembled components.

4.3.1 Service Component Architecture:

SOA requires appropriate techniques and tools for the delivery, support, and management of distributed applications conforming to its principles (Seinturier *et al*, 2009). SCA aims to fulfil that need with a specification for developing SOA applications that are technology agnostic. Pieber and Spoerk (2008) state that SCA provides a programming model for building SOA-based applications and solution which can run on a number of machines. The SCA programming model provides a technology-agnostic assembly capability for composing applications using available business services. SCA as an approach to developing SOA based applications assembles software components that use services. It is technology independent, and supports distributed configurations where remote components are interconnected by various means. Several infrastructures have been developed that implement the SCA specifications such as Apache Tuscany (http://tuscany.apache.org/) and Fabric3 (http://www.fabric3.org/) to be used as they are or customised to meet the developers' requirements.

The greater benefit of using SCA is that it builds distributed applications using SOA whilst applying CBSE principles. This makes it even easier to use by a novice in SOA with strong background in CBSE. The SOC paradigm provides a way for exposing coarse-grained and loosely coupled services that can be accessible remotely. CBSE provides the mechanism for defining the assembly of components. The shortcoming that SOC has is that it does not address the issues related to the real implementation of these services. As stated by Laws *et al*, (2011) even though SOA in an interesting idea, putting it into action can be a daunting task as business environments typically contain many different technologies leading to what might be a complex task to integrate. This then leaves SCA to address this shortcoming by defining a component model for these SOA applications. SCA gives developers an opportunity to focus on the business logic with no consideration of infrastructural details when building software applications.

There are technologies that have embraced SOA at the edges of application domain such as Web Services and ESBs with their proven methods. They have delivered key benefits in interoperability, governance and productivity. SCA provides the means to explore the same benefits without introducing a full Web Services stack for each interaction. Rather SCA uses SCA Bindings to isolate the specifics on how application components are both invoked and the way they communicate. SCA entities are software components which may be providing interfaces called services, may be requiring interfaces called references and exposing properties (Seinturier *et al*, 2009). It possesses well defined interfaces that define how some functionality can be provided. In SCA an application is broken down into a set of well-defined services to reduce the complexity and easing maintenance issues as isolated business functions are brought together. The main focus of the SCA specifications is the implementation of components in the service oriented environment emphasising a correlation between the existing software infrastructures or components with the newly created ones (Seinturier *et al*, 2012).

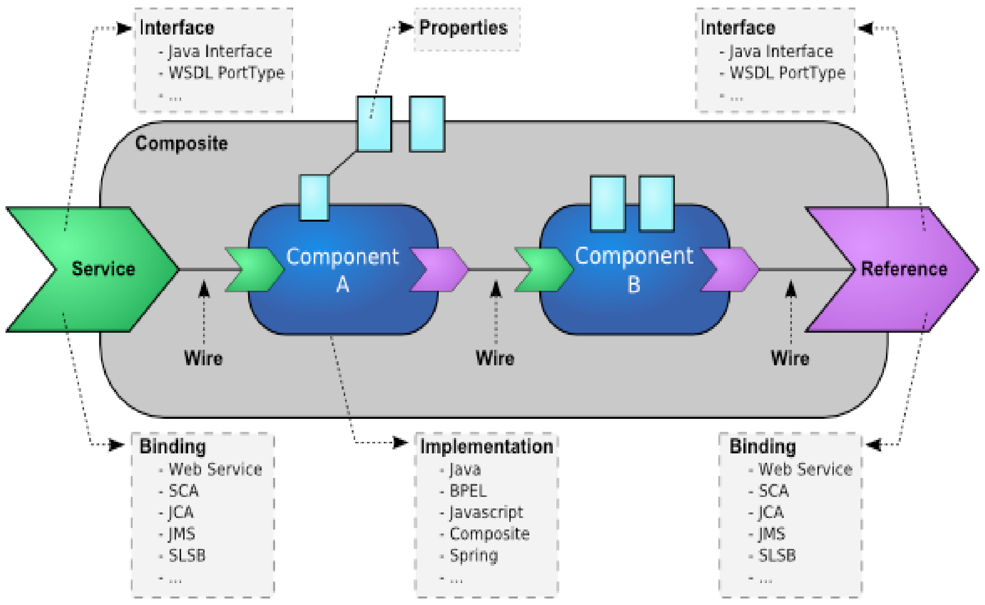


Figure 4.2 Typical SCA diagram (Chapell, 2005)

Figure 4.2 presents a typical SCA diagram with the relevant information on the resources that are combined to realise some composite. The basic building blocks for SCA applications are components, used by the unit of deployment for SCA known as a composite; it holds services which can be accessed remotely. A composite contains one or even more components containing the business function(s) provided by the modules. The Assembly Model is the core of SCA as it provides a clear separation between the business logic and other infrastructural issues. This assembly approach allows components implemented in some technology to be connected with components implemented in some other technology. The good thing about SCA components is that they all have the same architecture without the knowledge of each other's implementation type; this makes it easier to integrate them into a composite application. The idea is on having a component implementation, then using SCA bindings to expose the services it offers over multiple protocols.

Seinturier *et al*, (2009) defined these four principles underlying design of an SCA application. These principles are the building block for creating a service architecture that is as independent as possible from the implementation technologies in use.

- *Independence from programming languages*: SCA does not assume any programming language was used in implementing a component instead it supports several language mappings. This allows SCA components to be implemented using any of these languages Java, C++, PHP, BPEL or COBOL.
- *Independence from interface definition languages*: SCA components use interfaces to provide its functionality. SCA supports several interface definition languages (IDL) such as WSDL and Java interfaces. This enables interfaces built from any type of technology to be used.
- *Independence from communication protocols*: Even though web services are usually the preferred method of communication for SCA components, there is still a need for providing for other communication protocols where SCA components fall short.
- *Independence from non-functional properties*: Non-functional properties of an SCA component do not affect its business logic. A set of policies are declared to provide for the non-functional properties that a component depends on. Security is regarded as a non-functional property of the service since a service can fully function without security being applied.

These principles offer a broad scope of solutions for implementing SOA based applications using the SCA approach; in turn they form the basis for the deployment of the GUISET infrastructure as the most suitable approach. The GUISET infrastructure is and should be a flexible and robust infrastructure that handles the challenges of heterogeneity, dynamism and openness inherent in grid systems (Iyilade *et al*, 2009). Modern SOA applications should adapt to changing environments, support online evolution and be deployed dynamically. SCA provides such an environment except for dynamic adaptation in changing environments (Li and Parashar, 2006; Sienturier *et al*, 2009) which is the biggest challenge facing SCA specifications, basically defined as a lack of support for runtime management. To make SCA exhibit that much needed dynamism, it is combined with OSGi.

4.3.2 OSGi:

In order to meet the dynamic challenge discussed above, the OSGi service platform is configured to work together with SCA. OSGi is a dynamic modular system for Java enabling services used by the components to join and leave the environment on the fly without disrupting the functioning of the whole system. In dynamic deployment only the re-deployed components are stopped whilst the target application continues to be executing without interruptions (Ketfi and Belkhathir, 2005). The features that Java possesses enable support for products on many different platforms. Originally OSGi was introduced to implement component systems in Java that can be easily deployed into a Java Virtual Machine (JVM). The OSGi technology provides the standardised primitives that allow applications to be constructed using small, reusable and collaborative components which can then be composed together to form an application and deployed.

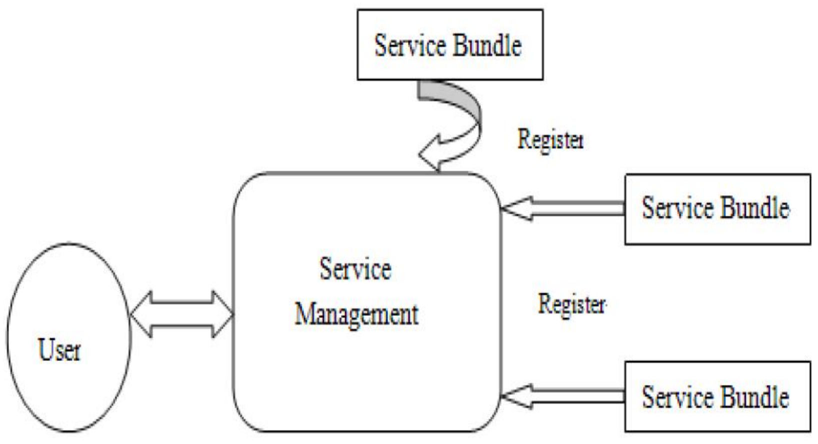


Figure 4.3: OSGI framework (Li *et al*, 2009)

Figure 4.3 depicts the OSGi framework showing the components that interact in the dynamic management of services. The service management component handles the introduction as well as the exiting of a service dynamically without disturbing other services that are not affected. With the growing success in large scale systems over the years it should still be acknowledged that OSGi was initially targeted for embedded applications such as smart appliances and mobile devices during its inception. In OSGi, software can be broken down into different modules called bundles. The core of the OSGi service platform is the framework or container responsible for creating a runtime environment for managing the deployment and lifecycle of bundles (Brada, 2008). Module systems provide version support for distributed bundles where a bundle goes beyond just an OSGi bundle. Modules subdivide a system into smaller parts that can be created and used independently in one or more systems to achieve some functionality.

Bundles as explained by Wu *et al* (2007) are libraries or applications that can dynamically discover other services from the service directory or can be used by other bundles. Simple Java classes can be regarded as bundles given that they are deployed to an OSGi runtime as JAR or WAR files to perform some functionality. Basically the physical form of a bundle can be simply stated as a JAR archive with a manifest file that contains metadata defining the bundle and its features. Eclipse Equinox is an implementation of the OSGi R4 Core Framework specifications. It simplifies the process of developing and deploying modern software applications by providing a lightweight component-based method of building applications. Earlier Equinox was used to simplify the integration and the development of developer-oriented tools before being used as a runtime for running applications from different platforms.

Combining SCA and OSGi to realise an infrastructure theoretically is easy but in practice it somewhat brings some challenges; such as finding a binding that enable the use of services implemented in any technology. Apache Tuscany comes packaged with the Jetty Server which at some instances fails in finding the deployed services. The advantage is that Apache Tuscany gives opportunity to extend its capabilities through pluggable components. The

OSGi runtime Apache Karaf becomes lightweight because of its opportunity to allow the developers to only implement the functionalities they need from it.

4.4 Introduction to Open Source solutions:

Over the years open-source software has been getting a lot of attention from both academics and the business world due to the successes it is getting in the market. Open-source solutions are freely distributable; come packaged with the code to enable the extension of capabilities to meet the preferred requirements. Just to mention a few open-source solutions that are currently widely used in the market and academic institutions include Linux, MySQL, Mozilla Firefox and Apache software products. Even though open-source is freely distributable it is not necessarily free, meaning that one might have to pay to have rights to use it (Haapasalo, 2007; Maican, 2009).

There are so many beliefs regarding the untrustworthiness of using open-software compared to proprietary software (Haapasalo, 2007). The fears surrounding open-software are that it lacks quality, not enough documentation and above all there is the fear that open-source software lacks support and maintenance (Haapasalo, 2007). All these fears have not stopped open-software from gaining the exposure needed to be used especially due to fewer limitations in vendor lock in and the cost of ownership. Most developers use open-source solutions because it is extensively available in the sense that it is free compared to its commercial counterpart solutions. The next sub sections introduces the open source solutions applied in this work, in Section 4.4.1 SCA: Apache Tuscany is introduced and in Section 4.4.2 OSGi: Eclipse Equinox is introduced

4.4.1 SCA: Apache Tuscany:

In the development of SOA application based on SCA specifications two platforms standout based on the literature that has been explored. Those are Apache Tuscany and Fabric3. Apache Tuscany and Fabric3 are somewhat similar so due to the availability of supporting documentation, user support and popularity of Apache projects the selection of Apache Tuscany became relevant for this project. Apache Tuscany is an open-source solution developed using the SCA specifications by the Apache Software Foundation. The Tuscany runtime provides a modular and pluggable architecture for developers to pick and choose their intended functionality and discard what is not needed. The Assembly Model of the SCA specifications simplifies the development of the application by separating infrastructure concerns from the business logic. The following are the capabilities that Tuscany provides to address the complexity of SOA applications:

- Choose the language used to implement components.
- Assemble components implemented using different technologies into composite applications.
- Configure communication protocols with no modification to component implementations.
- Control non-functional application behaviour using policy configuration.

These capabilities are achievable using existing technologies with the Assembly Model describing how these technologies will come together to form a working application. Tuscany can be embedded with other application containers such as Java EE or OSGi based containers to give you that flexibility to run your application in the environment of choice. As the shortcoming of SCA specifications was already stated lack of dynamism it’s the same for Tuscany as the tool that implements SCA.

4.4.2 OSGi: Eclipse Equinox:

To develop software applications that dynamically use the services defined by the assemblies created through Apache Tuscany, there is a need to integrate Apache Tuscany with one possible OSGi runtime to maintain dynamism. The possible candidate OSGi runtimes that were being considered in this were Apache Karaf, Eclipse Equinox and Apache Felix. Based on the reported work there is not much difference in the usage of any of these OSGi runtimes, they all come as standalones and can be plugged in to some other tool. The choice of using Eclipse Equinox in this instance was left to the developer who deemed its immediate ease to be plugged into the Eclipse IDE as the selection factor and also its capability of only picking and choosing the functionalities that are expected of it. The Eclipse Equinox is dynamic runtime implementation based on the OSGi R4 Core framework specification by the Eclipse community. The Eclipse Equinox implements all aspects of the OSGi specifications that is inclusive of the mobile, home and vehicles aspects.

The discussed set of open-source solutions has complemented each other in the development of the GUISET infrastructure. It should be noted that there is not strict adherence to the selected tools but rather the focus was on selecting the paradigm that is suitable and then the tools rely on that selection of the paradigm. The next section looks at how both the concept of SCA and OSGi has been applied in the reported work.

4.5 State of the SCA and OSGi runtime environments:

This section presents the current state of the art and a justification for the adoption of SCA and OSGi framework runtime environment. Several works have been realised with each aiming to enable the assembly of business services dynamicity in computing environments and some special features based on the research agenda that was addressed. For the purposes of this work, the selection of a suitable mechanism to address service composition and enablement of a dynamic runtime environment was based on the reported information on the capabilities of the each of these approaches and informed by the characteristics of GUISET. The assembled components must always find services to use at runtime.

Huang *et a*, (2011) implemented an e-Tourism system based on SCA with the intention of providing full services for tourists and decision support for tourism managers. The core of e-Tourism functionalities is based on SOA and ESB, with SOA implemented using SCA specification for providing business logic and ESB establishing communication. The e-Tourism is developed using open-source solutions, with Mule ESB implementing the integration of services used with the Spring runtime framework handling the dynamicity of the system.

Dorminger (2009) presents the design, implementation and deployment of a runtime environment based on the Eclipse Equinox OSGi framework. This research work builds a process model called ProMoRTE using some existing components together with components built in-house. ProMoRTE is simply defined as a general purpose runtime environment for executing computational algorithms based on Java and the OSGi framework. The flexibility that Equinox brings in this project is its ability to dynamically manage components and their versioning support on the fly, and also a core requirement for the GUISET infrastructure as it has to manage the assembled components on the fly.

Based on the work that has been reported here it has been realised that the Apache Tuscany which is based on SCA specifications and the Eclipse Equinox which is based on OSGi are the most optimal solutions for adoption in the development of the GUISET infrastructure. These two concepts complement each other in that the SCA specifications address the assembly of software components to address some business process.

4.6 Chapter Summary:

This chapter started by presenting GuSCaDA, the architecture that has been formulated to realise an infrastructure that supports both service composition and dynamic service binding. Then the concept of software development and evolution together with SOC concepts that achieve it for development of a GUISET infrastructure that is evolvable over time was presented. This was then followed by the open-source solutions that have been applied in the realisation of the GUISET infrastructure. Lastly, the state of the art of SCA and OSGi was introduced to shed light on research activities that have been conducted and how far the study has been conducted.

Chapter 5: Prototyping a GUISET SMME enabler

5.1 Introduction:

This chapter presents the realisation of both the GUISET infrastructure and NOS application. The implementation of these two prototypes, the GUISET infrastructure and the NOS application was intended to show how (1) the infrastructure was realised using available tools and (2) validating the infrastructure by deploying an application it should support. For purposes of this work only the first GUISET prototype was developed to act as infrastructure that can be used by NOS application to achieve its business goal. Then the prototyping concept as an approach to the development of these prototypes is also presented with its different approaches and the justification of selecting the evolutionary approach to allow different iterations of the prototype to be produced to cater for the requirements of GUISET, both old and new. Then, the UML diagrams are then presented to outline how both the infrastructure and the NOS application have been modelled.

This chapter began by presenting the assumptions that have been made on each prototype to highlight what has been covered in the implementation of the prototype. The intention is realizing the use cases in Section 5.2. This is then followed by Section 5.3 introducing the prototyping concept and its various approaches in the realisation of a project including the approach selected for this work. Finally, in Section 5.4 and Section 5.5 the GUISET infrastructure and NOS application prototypes are presented respectively.

5.2 Prototypes Assumptions: GUISET infrastructure and NOS Application:

Both the GUISET infrastructure and the NOS application have been realised with the intentions of proving the work reported here. This made the focus to deter from realising a full infrastructure and an e-business application with most of the regular functionalities. In this work, the infrastructure using the open-source solutions were put together to realised an infrastructure that supports both service composition and dynamic service. The NOS application only queried the catalogue and placed an order for a product. The concentration was only on achieving service composition and dynamic service binding. The infrastructure provides an environment for the NOS application that comes with a clearly defined assembly of components based on its application scenario as this remains static, dynamicity is then used for binding to services that closely addresses the business goal.

This enables the prototype to undergo a series of improvements towards meeting the intended requirements. Prototyping provides advantages such as early feedback in the development process, fast time to market, and these two were the most notable reasons for adoption in this work. There are several prototyping approaches that can be applied and it is not easy to say which prototyping approach is best from face value without direction towards the project requirements where is it is applied. The next section presents prototyping as a concept as its approaches.

5.3 Prototyping:

Prototyping or Software Prototyping refers to the rapid software or system development to validate requirements and is the process of utilising prototypes in software or system design (Sommerville, 2001). The purpose of a prototype is to model a system, or specific aspect of the system (SQA FIVT 34, 2011) in an iterative manner to get a working prototype (Farrel, 2007). Using the prototyping method a project is broken down into small prototypes that are implemented independently of the entire system. This makes the development a lot easier as concentration is only delegated to that aspect being developed.

Software prototyping enables the building of a software application through a set of prototypes. These prototypes are put together to realise the intended functionalities of the whole system. The principal use of prototypes is to help customers and developers understand the requirements of the system by enabling early feedback. In turn it gives opportunity to analyse and refine the systems intended requirements that must be met. Frequent prototypes are produced and analysed to enable the functionality of the system to be better understood until what is perceived to be the intended outcome is achieved. In this form of development, ambiguities are eliminated from the system requirements as it entails more interaction between the developers and the users of the system and with each prototype being developed early feedback is provided to give direction as to how it performs in addressing the identified requirements.

Prototyping as a methodology uses a number of methods which can be sequentially grouped according to their execution sequence. Farrel (2007) states one advantage of prototyping as a method that provides a view into the product functionality and its usability early and throughout the production process with the view of the changes that occur. Prototyping practices these three types of approaches namely; throw away, incremental and evolutionary prototyping all with their strengths and weaknesses. The following section presents the clear distinguishing factors among these prototyping approaches. The aim is to lay a platform for the selection of the most appropriate approach. The system requirements are usually the main determining factor in the selection of the approach to be used rather than the preference for the approach by developers. The following subsections, Section 5.3.1 and Section 5.3.2 present the prototyping approaches and the selected approach respectively.

5.3.1 Prototyping Approaches:

Throw Away / Rapid Prototyping: throw away prototypes are used to merely gain knowledge and/or inform the system requirements. In the development of a software artefact throw away prototype usage goes as far as testing the validity of some aspect without featuring in the actual realisation. This approach is also referred to as rapid prototyping due to its ability to quickly build something to get feedback or eliminating some ambiguity.

Incremental Prototyping: the building of the software artefact allows for the development of independent usable components. Components built satisfy some aspects of the system, and the mock up graphical user interface will generally expose some functions that have not been implemented. This approach gives opportunity for integrating components that implement the previously unimplemented functions to the system without affecting the already existing components.

Evolutionary Prototyping: the evolutionary approach aims to develop a mature system through a series of prototype iterations. The prototype will undergo a series of refinements, and should eventually become the final solution or it can be better explained as more like developing different versions of the system with each version having addressed some drawbacks that have not been addressed in the previous versions.

5.3.2 Selected Approach:

GUISET aims to provide an infrastructure for SMMEs as an IT enabler. The technical requirements can be summed up as an infrastructure that is evolvable over time to provide for users' future requirements. Since the future cannot be foretold, evolutionary prototyping is explored and applied in this work. The primary aim of applying the evolutionary prototyping approach is to put together a series of prototypes to realise an infrastructure that can be continuously reviewed and updated, producing different versions that will be addressing some additional user requirements. The work reported here though is limited to producing the first GUISET prototype. The GUISET prototype produced here is only concentrating on addressing service composition and dynamic service binding. Only future work will produce further iterations of the prototype that will be evaluated and improved to meet the GUISET infrastructural requirements. With GUISET envisioned to being a multipurpose infrastructure, only the e-business scenario is addressed in this work. The use of the Nongoma Online Stores an online application was deemed as an evaluation strategy for this work. The completion of an order for a product using the NOS guarantees that the GUISET infrastructure is provided the intended support. Later, other modules such as a health, booking and other modules are expected to be effected with no/not much disruption on the functionality of the infrastructure.

5.4 Prototyping the GUISET Infrastructure:

The implementation of the GUISET infrastructure is based on the proposed architecture of Figure 5.2 that presents a more technical view of infrastructure based on the technologies applicable. This (Figure 5.2) is informed by the conceptual architecture of Figure 5.1 in the beginning of this chapter which gives a basic idea of the interacting components to realise the infrastructure.

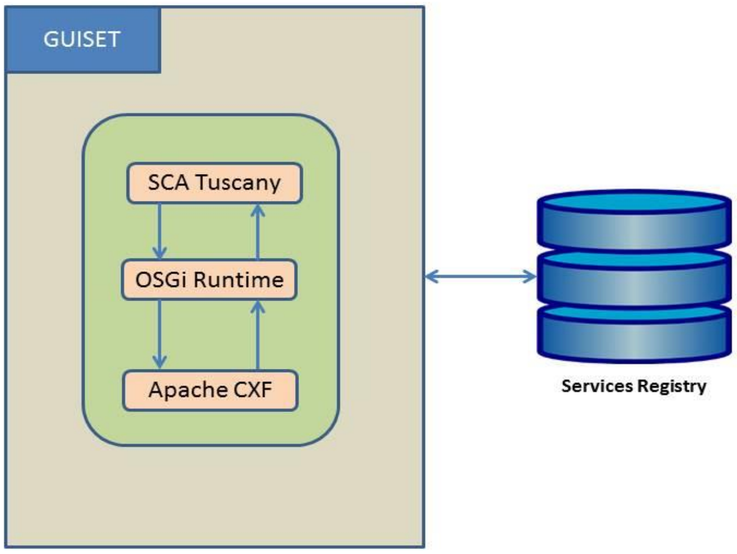


Figure 5.1: GUISET Middleware Interaction Architecture

Figure 5.1 depicts a closer look at the middleware layer showing how a combination of computing resources and techniques has been applied in achieving the GUISET infrastructure. In light of the GUISET infrastructure supporting service composition and dynamicity which are open research issues, this work does not address problems with regard to those open issues but rather the combination of tools that realises these research areas are adopted and used in this work. This work, in the prototyping of the infrastructure, assumes there are no current issues. The tool selection is based on the characterisation of GUISET. As mentioned earlier an application that will successfully be supported by the infrastructure will give enough evidence to declare it as a GUISET infrastructure. The success of the GUISET infrastructure is achievable through the combination of Apache CXF, OSGi Runtime and SCA Tuscany each addressing some aspect as described in the following:

- SCA Tuscany: provides a comprehensive infrastructure for the development and management of SOA applications based on SCA specifications. Apache Tuscany provides a lightweight runtime with a modular and pluggable architecture so users can choose the functionality that they need and discards all the rest. To make Tuscany achieve our main goals it has been used as an Eclipse plugin to enable extensibility.
- OSGi Runtime: was selected to handle the hot deployment of components and version support. Its ability to be integrated with the Eclipse IDE makes this a more advantageous runtime to utilise.
- Apache CXF: is an open source services wrapper it helps to build and develop services using frontend programming APIs, like JAX-WS and JAX-RS or even to be used as a binding to other platforms. These services can interact with a variety of protocols such as SOAP, XML/HTTP, RESTful HTTP, or CORBA and work over a variety of transports such as HTTP, JMS or JBI.

To best address the combination the presented tools SCA Tuscany, OSGi Runtime, Apache CXF in the realisation of GUISET infrastructure and the supporting application prototyping approaches were observed to understand which one may best suit this project. The basic motivator for the selection of prototyping as an approach was basically its ability to provide early feedbacks in the development giving opportunity for evaluating the progress of the project and possibly addressing matters of concern early in the stages of development.

The implementation requires these three steps. First, components are assembled based on some business goal that must be achieved. Next, there is a need for a technique that will enable the dynamic binding to the services defined in the assembled components in the first step. Finally, external services to be used may be implemented in any technology. There is a need for a mechanism that manages the differing technology protocols to use services implemented in any technology.

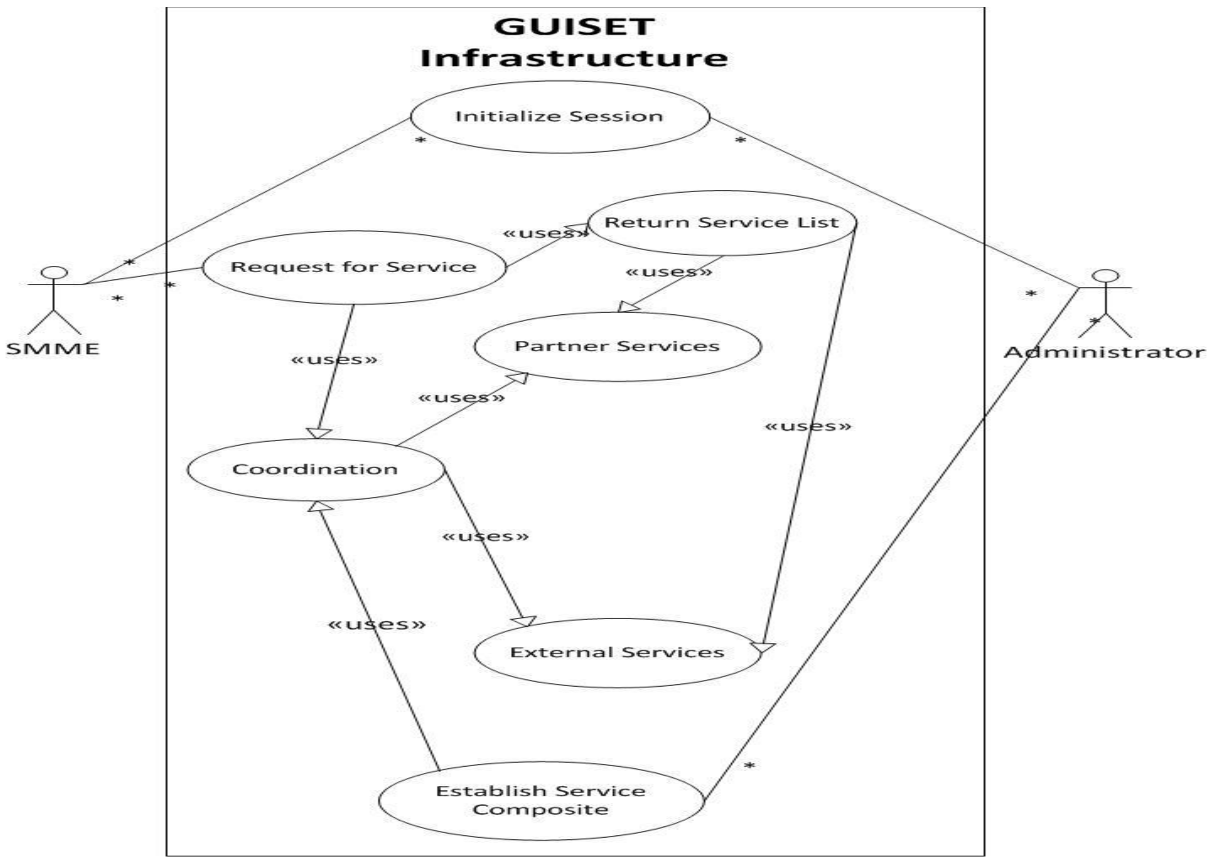


Figure 5.2: GUISET Infrastructure Use Case Diagram

The use case diagram in Figure 5.2 represents the interaction between the user and infrastructure. The user interacts with the GUISET interface which in turn uses the Coordination to assemble services found in the Partner Services. Partner Services use External Services to provide the necessary functionality.

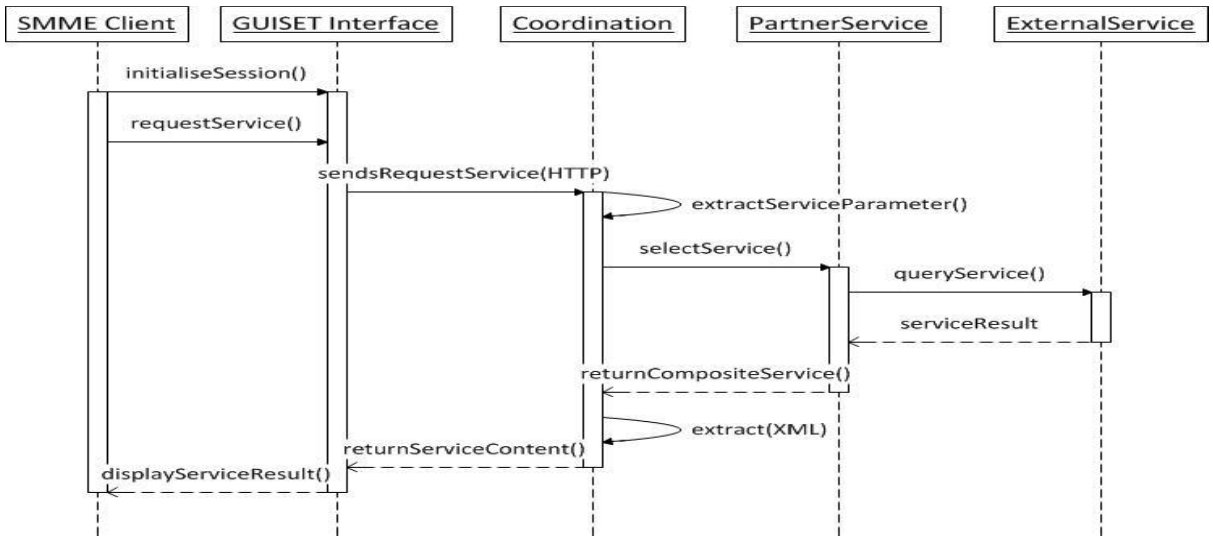


Figure 5.3: GUISET infrastructure sequence diagram

The sequence diagram in Figure 5.3 shows the sequence of events among the interacting components that make up the GUISET infrastructure. The flow of messages is initiated by the user requesting services for managing his/her business processes. The SMME client is the initiator of any session or business process that must be handled by the GUISET infrastructure. The SMME client initialises the session through an interaction with the GUISET interface. The GUISET interface receives a request for a service which in turn is sent to the Coordination component. The Coordination component first extracts the service parameters before selecting the relevant service from the Partner services. Once the Partner services receive the request it queries the external services repository for a composite of services that fulfil the request. External services are the outsourced services providing some functionality at a given time. The essence of using external services is that there should always be an available service to satisfy users' requests.

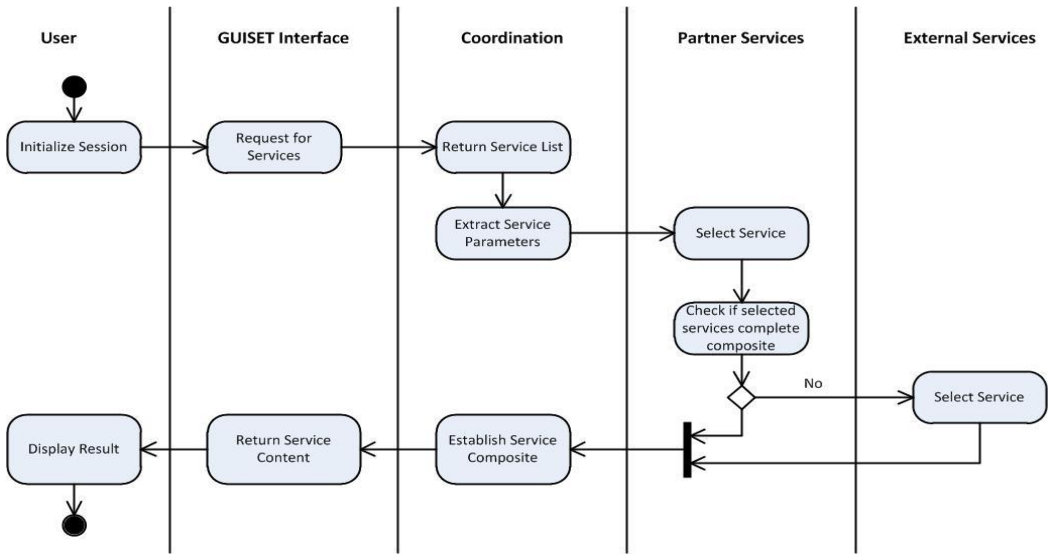


Figure 5.4: GUISET Infrastructure Activity Diagram

The activity diagram in Figure 5.4 shows the flow of activities that take place when the infrastructure is in operation. Assumptions that have been established pertaining to this infrastructure is that services are always available. That is why no decisions were taken should the service be not available. The activities start when the user initialises a session which is done simultaneously with the request for a service. The request is to the Coordination to assemble the services that are needed to complete the process. Once the service request is sent, Partner services receive a query to establish the composite of service then finally the results are returned to the user.

5.5 Prototyping the NOS application:

The implementation intends to illustrate how the GUISET infrastructure has been realised through the use of existing tools. The NOS application on completion of its business process will validate the GUISET infrastructure through the application of service composition and dynamicity service binding for the enablement of the sale of some art and craft product or some other process given the infrastructure is in support of completely different applications. For the purposes of the NOS application enablement, the SMME as the client of GUISET infrastructure registers its process prior to its usage by defining abstract services, the abstract service being interfaces defining the functionality. The customer buying products from the NOS triggers a process which in turn uses the abstract services as guidance to which concrete services to use, concrete services being service implementations that provide the actual functionality.

The Nongoma Online Stores (NOS) is a typical implementation of an application that is supported by the GUISET infrastructure for selling arts and crafts, with the infrastructure expected to feature a variety of applications with differing functionalities and goals. For the purposes of this work, the NOS application has not been implemented to perform as many functionalities as one would expect in an online store. The implemented functionalities are the querying of the catalogue and making an order for available products. This is to show how the components have been assembled and to achieve dynamic binding to services. Following is both a SCA design and a graphical view of the service interactions.

The diagram represented by Figure 5.5 gives an abstract view of how the interaction of components occurs between the infrastructure, the supported application and service providers in realising its goal. For clarification purposes in the figure the design has been numbered with (1) representing the supported application which has been termed full-app user interface (UI) in this case being the NOS application, (2) representing the infrastructure which has been termed coordination UI, (3) being the providers of concrete services being termed partners UI and (4) being the cart for temporary storage during the session being termed shopping-cart UI. The module for placing orders for the items on request has been implemented in this work as the realisation of the arts store and played a role addressing the arguments made throughout this work such as service composition, dynamic service binding, software development and evolution and usage of SCA specifications to outline a few.

NOS and Infrastructure SCA-based Component Designs:

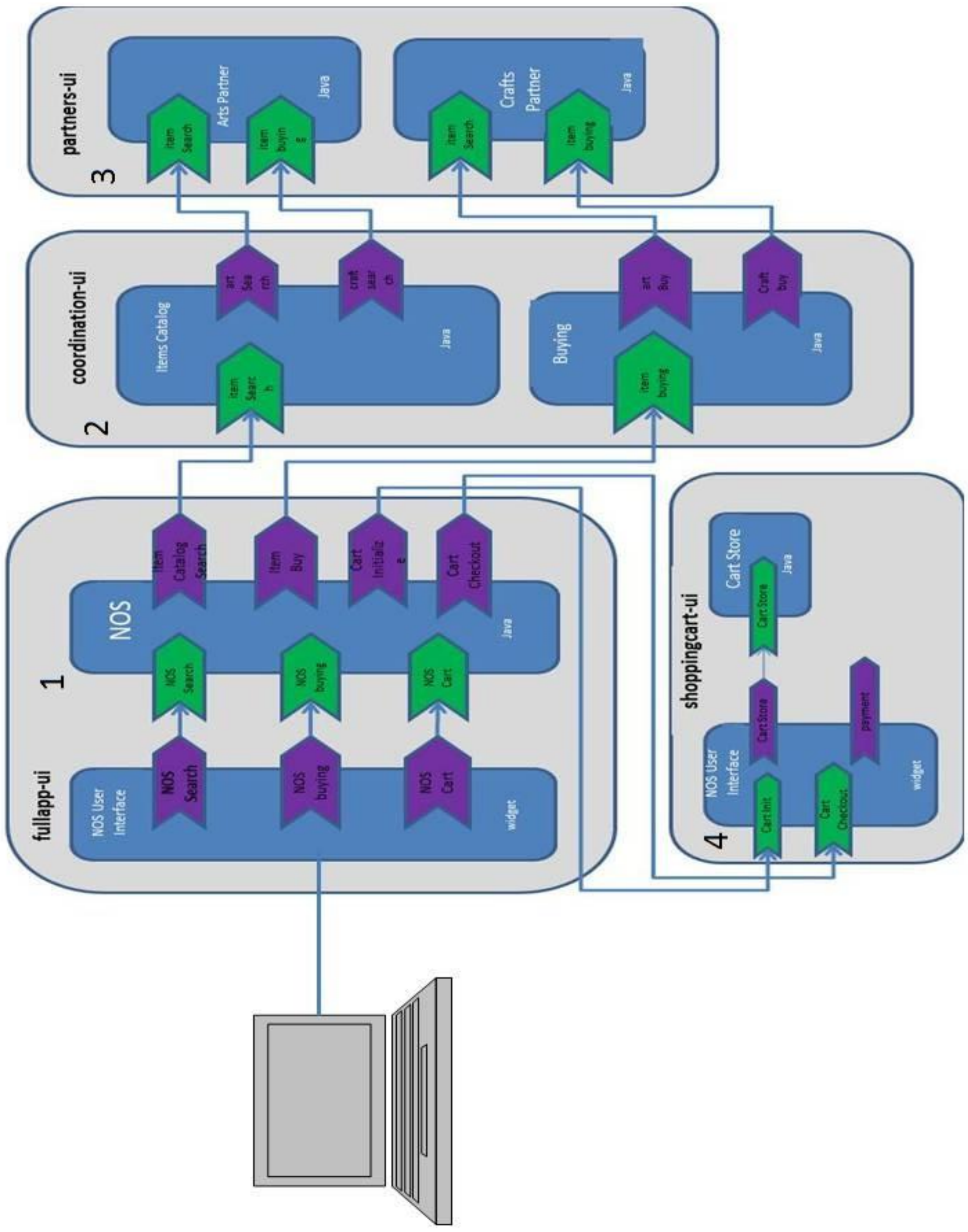


Figure 5.5: Nongoma Online Store and Infrastructure SCA-based designs

The NOS and infrastructure SCA-based design has four separate composites. These composites are independent of one another in terms of performing the intended business activities. Each composite is made up of one or more components using services. The names and functionality of each composite is defined as follows:

Full-app-ui: acts a front-end to the user of the system, it contains all the functions that the system performs, such as searching the catalogue, buying products and managing shopping cart. Widget has been used for the interface interacting with the user and the business logic is implemented using Java.

Coordination-ui: implements the core business logic of the system. It consists of two components, one for searching the catalogue for specific items and the other for buying items of choice. The components have been implemented using Java.

Partners-ui: contains third-party services that provide the necessary products. It is integrated with the coordination-ui to provide the available services relating to arts and crafts. These services are implemented in any technology since they are outsourced from outsiders.

Shopping-cart-ui: for managing the orders received by the system. A cart is initialised to be only available for that particular session and destroyed immediately when the transaction elapses. Each cart is managed using an id automatically created on initialisation

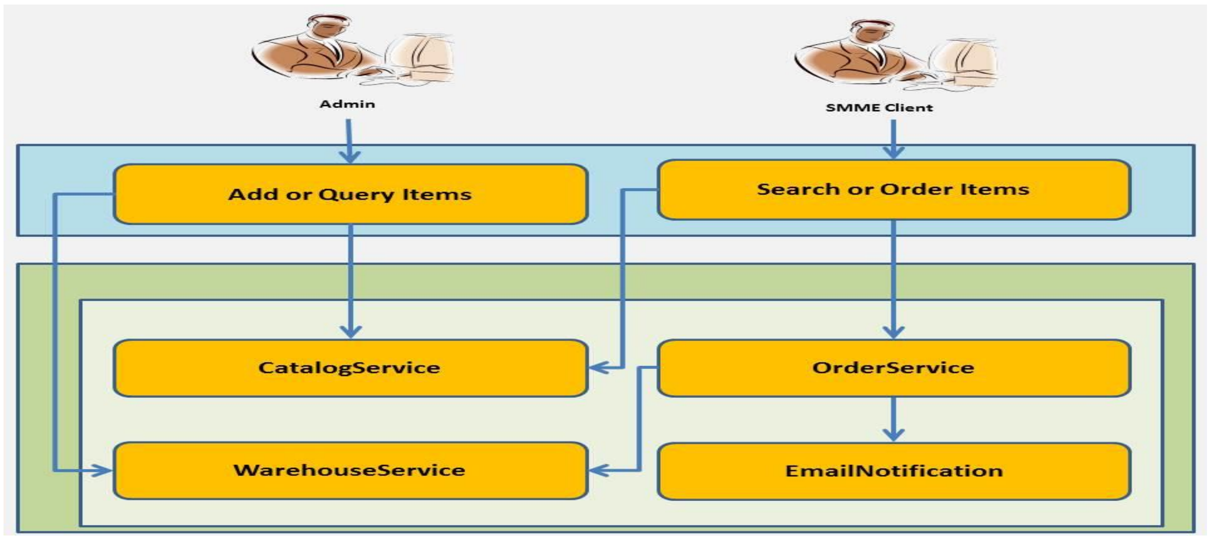


Figure 5.6: Service interaction diagram

Figure 5.6 shows an abstract view of the services interacting in processing a users' (admin or SMME client) request. The administrator is responsible for adding item details in the data warehouse and also querying for the available items, these use the CatalogService and WarehouseService respectively. The SMME client browses for available items and/or even makes an order for those items of interest. These use the CatalogService and OrderService respectively. In a case where an order has been placed using the OrderService, the OrderService will in turn reserve the item through the WarehouseService and an email notification will be sent via the EmailNotification service specifying the details of the possible buyer to the item seller.

Next, are the UML sequence diagrams depicting the business processes performed by the infrastructure. These UML diagrams show the interacting objects with the system actor issuing the request.

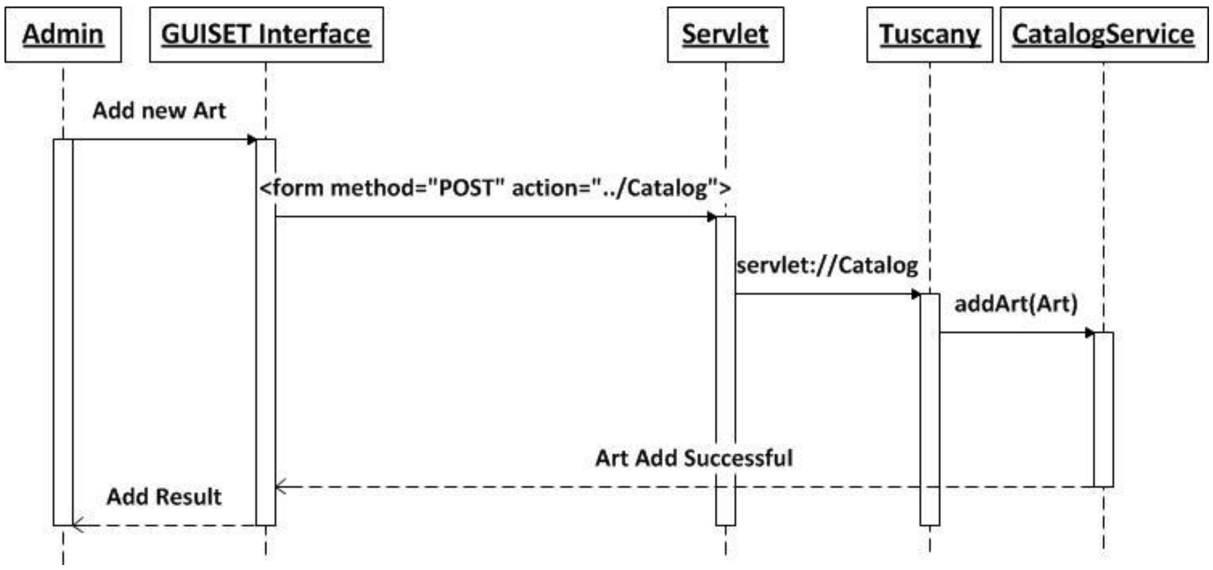


Figure 5.7: Administrator sequence diagram

Figure 5.8 represents a graphical view of the flow of events representing the process of adding an item. An administrator through the GUISET interface enters the item details such as the name, price and quantity. The form persists this information using a servlet through

Tuscany, Tuscany is responsible for transforming the HttpRequest into the Art object. On successful addition of an item the result is returned in the interface for the user to see.

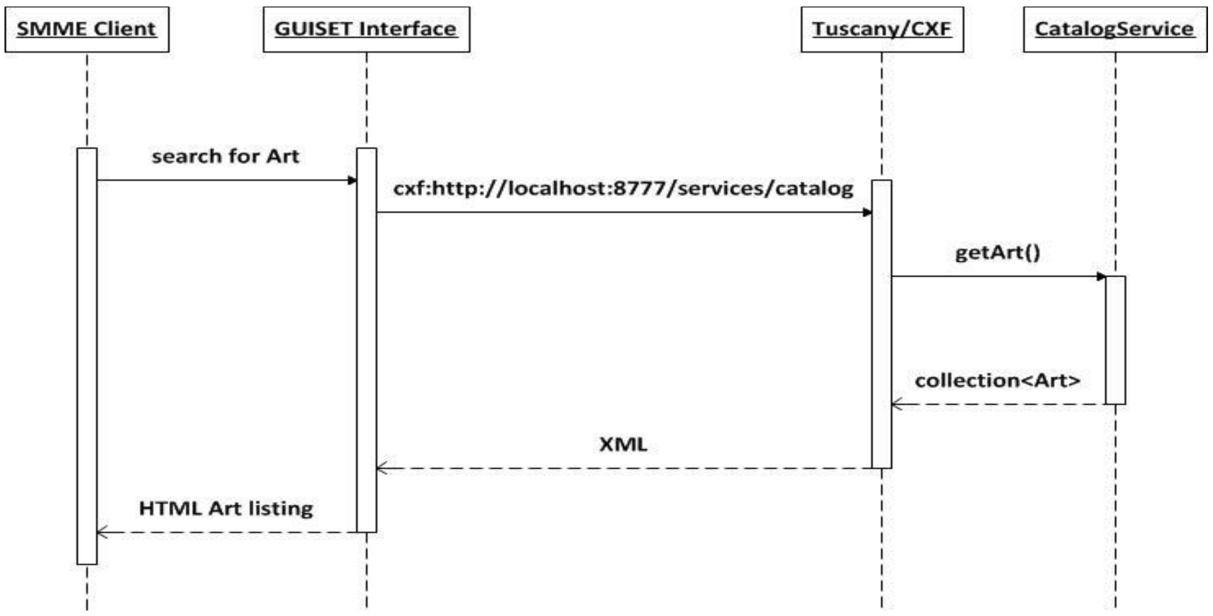
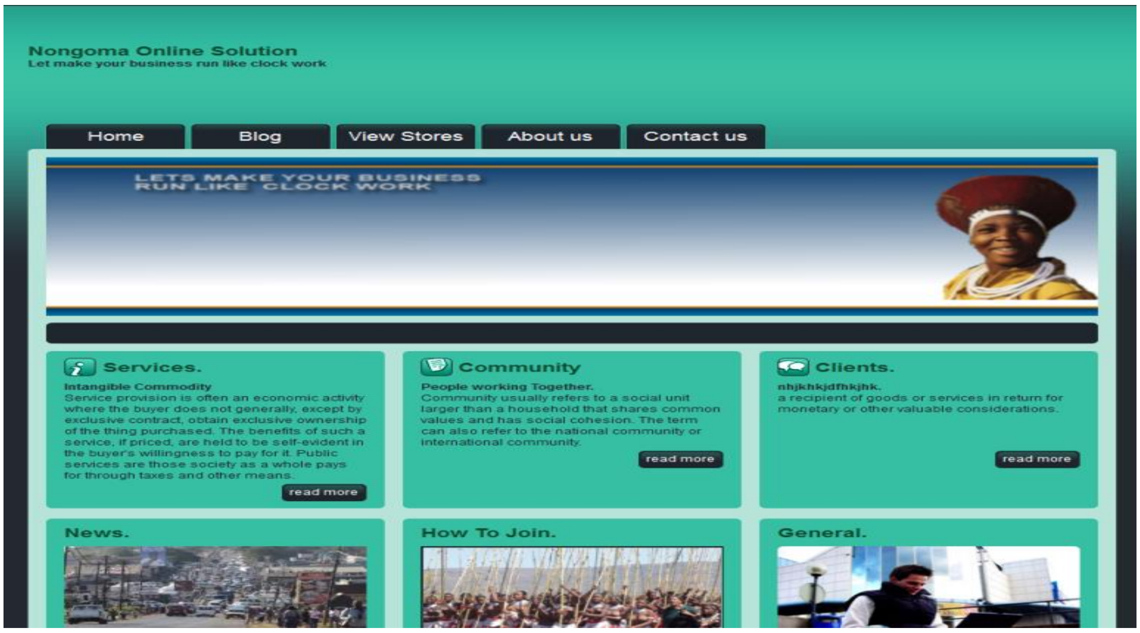


Figure 5.8: SMME Client sequence diagram

Figure 5.9 shows a graphical representation of the flow of events in processing a search through the catalogue. The SMME client using the GUISET interface enters the details of the item being queried. A binding CXF marshals the request between Java and WSDL/XML through Tuscany to communicate with the catalogue service querying the Art object. The Art object returns the results to the GUISET interface, which has been transformed by CXF binding in Tuscany. The SMME client views the result in an html format.

The following interfaces just show how the user interacts with NOS application in performing their intended transactions.


Nongoma Online Solution
Let make your business run like clock work

Home | Blog | View Stores | About us | Contact us

LETS MAKE YOUR BUSINESS RUN LIKE CLOCK WORK

Services.
Intangible Commodity
Service provision is often an economic activity where the buyer does not generally, except by exclusive contract, obtain exclusive ownership of the thing purchased. The benefits of such a service, if priced, are held to be self-evident in the buyer's willingness to pay for it. Public services are those society as a whole pays for through taxes and other means.
read more

Community
People working Together.
Community usually refers to a social unit larger than a household that shares common values and has social cohesion. The term can also refer to the national community or international community.
read more

Clients.
nhjkhkjdfhkjhk.
a recipient of goods or services in return for monetary or other valuable considerations.
read more

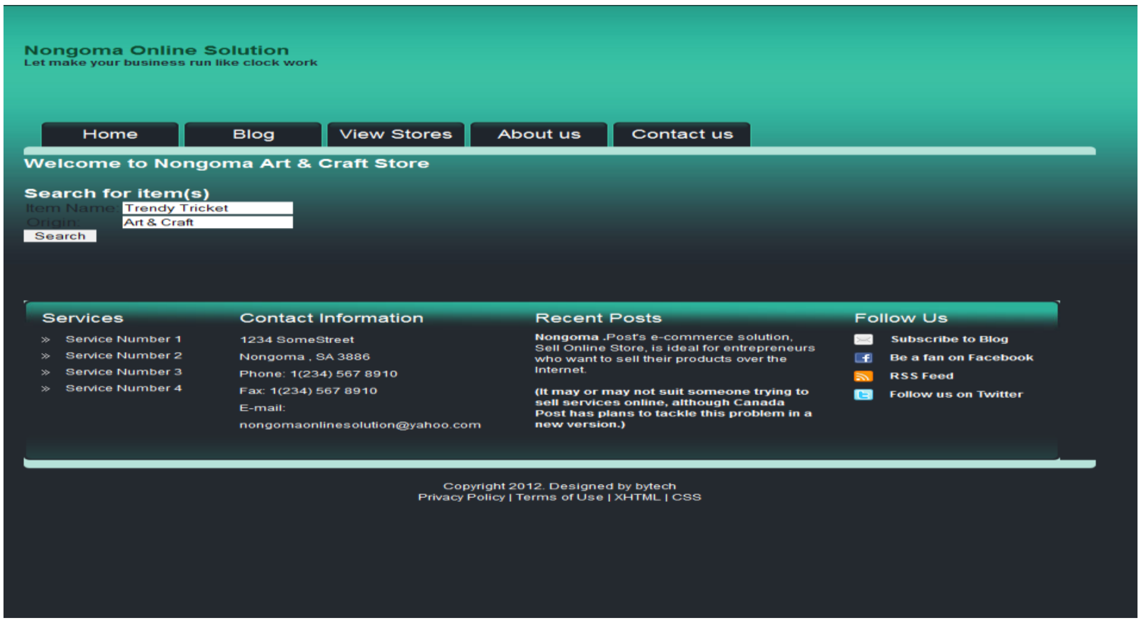
News.

How To Join.

General.

Figure 5.9 Nongoma Online Store Solution

This is the home page that acts as a starting point to the Nongoma Online Stores. It gives some idea of the services and other useful information of the Nongoma area. To start using the store, a simple click to the View Stores tab will display the store as per figure below.


Nongoma Online Solution
Let make your business run like clock work

Home | Blog | View Stores | About us | Contact us

Welcome to Nongoma Art & Craft Store

Search for item(s)
Item Name: Trendy Tricket
Origin: Art & Craft
Search

Services
- Service Number 1
- Service Number 2
- Service Number 3
- Service Number 4

Contact Information
1234 SomeStreet
Nongoma , SA 3886
Phone: 1(234) 567 8910
Fax: 1(234) 567 8910
E-mail:
nongomaonlinesolution@yahoo.com

Recent Posts
Nongoma .Post's e-commerce solution, Sell Online Store, is ideal for entrepreneurs who want to sell their products over the Internet.

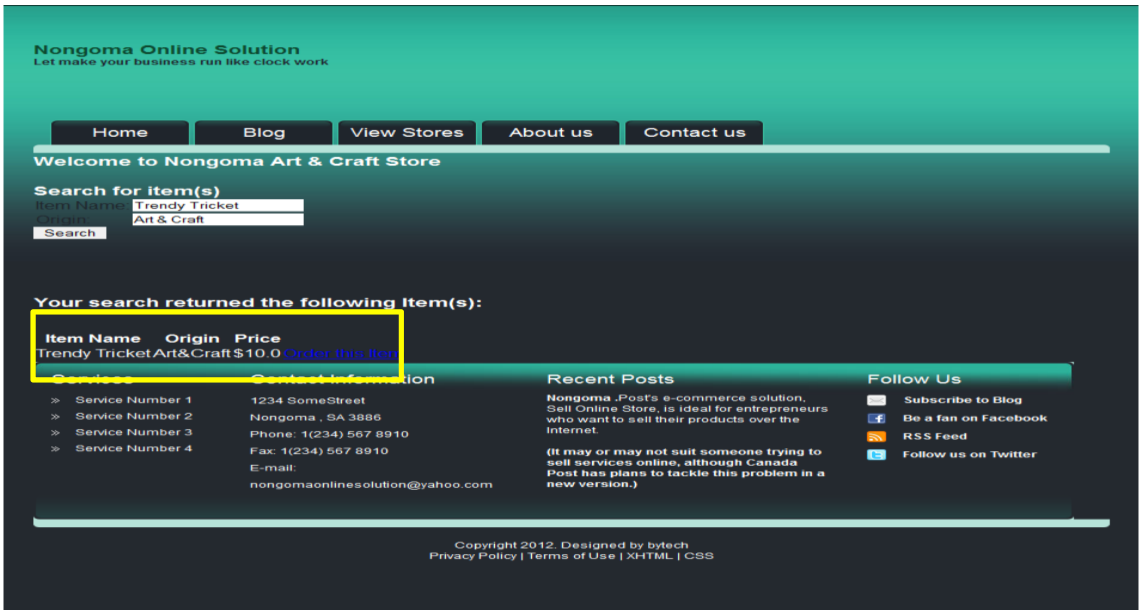
(It may or may not suit someone trying to sell services online, although Canada Post has plans to tackle this problem in a new version.)

Follow Us
Subscribe to Blog
Be a fan on Facebook
RSS Feed
Follow us on Twitter

Copyright 2012. Designed by bytech
Privacy Policy | Terms of Use | XHTML | CSS

Figure 5.10: Example of a search for a product (Trendy and Trinket)

This is the interface that the client queries for the item of choice. The client enters the item name the textbox then followed by a click on the search button in Figure 5.12.


Nongoma Online Solution
Let make your business run like clock work

Welcome to Nongoma Art & Craft Store

Search for item(s)
Item Name: Trendy Tricket
Origin: Art & Craft
Search

Your search returned the following Item(s):

Item Name	Origin	Price	
Trendy Tricket	Art&Craft	$10.0	Order this Item

Services
» Service Number 1
» Service Number 2
» Service Number 3
» Service Number 4

Contact Information
1234 SomeStreet
Nongoma , SA 3886
Phone: 1(234) 567 8910
Fax: 1(234) 567 8910
E-mail:
nongomaonlinesolution@yahoo.com

Recent Posts
Nongoma .Post's e-commerce solution, Sell Online Store, is ideal for entrepreneurs who want to sell their products over the Internet.

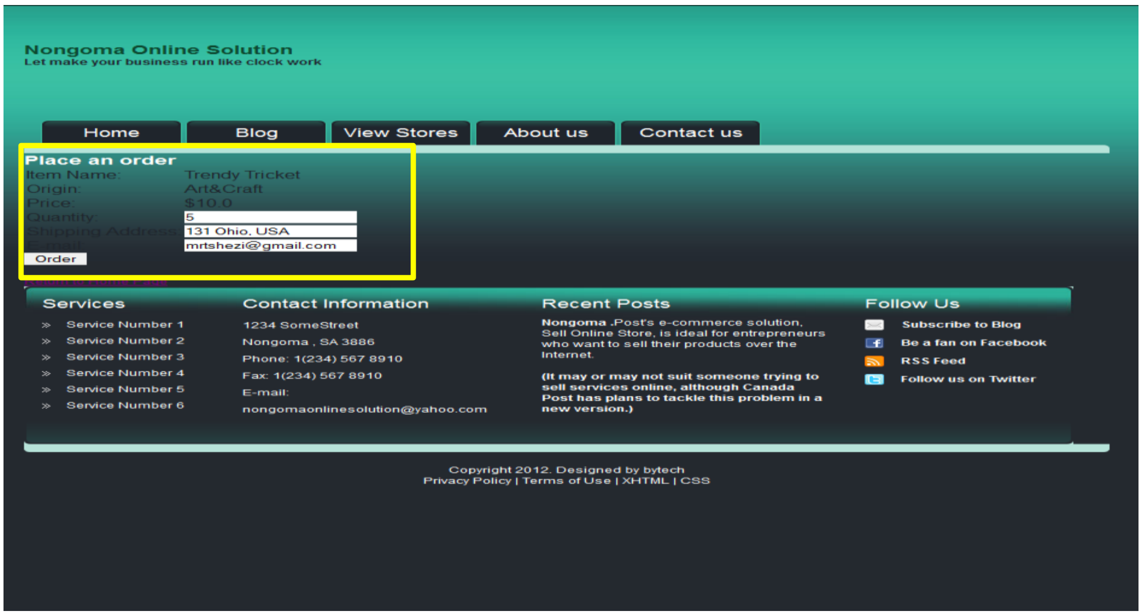
(It may or may not suit someone trying to sell services online, although Canada Post has plans to tackle this problem in a new version.)

Follow Us
Subscribe to Blog
Be a fan on Facebook
RSS Feed
Follow us on Twitter

Copyright 2012. Designed by bytech
Privacy Policy | Terms of Use | XHTML | CSS

Figure 5. 11: Example of the returned product (Trendy Trinket) after search.

This interface shows the results of the previous query for an item. The item search results are presented as shown inside the yellow box. To place an order for this item, the client clicks on Order this item shown in Figure 5.13.


Nongoma Online Solution
Let make your business run like clock work

Place an order

Item Name:	Trendy Tricket
Origin:	Art&Craft
Price:	$10.0
Quantity:	5
Shipping Address:	131 Ohio, USA
E-mail:	mrtshezi@gmail.com

Order

Services
» Service Number 1
» Service Number 2
» Service Number 3
» Service Number 4
» Service Number 5
» Service Number 6

Contact Information
1234 SomeStreet
Nongoma , SA 3886
Phone: 1(234) 567 8910
Fax: 1(234) 567 8910
E-mail:
nongomaonlinesolution@yahoo.com

Recent Posts
Nongoma .Post's e-commerce solution, Sell Online Store, is ideal for entrepreneurs who want to sell their products over the Internet.

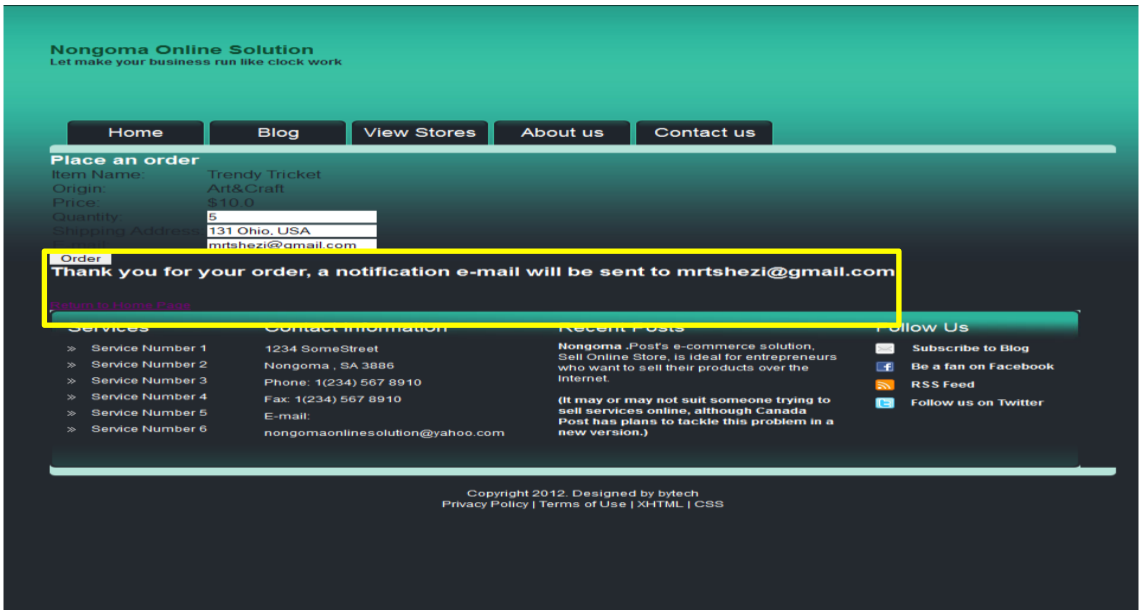
(It may or may not suit someone trying to sell services online, although Canada Post has plans to tackle this problem in a new version.)

Follow Us
Subscribe to Blog
Be a fan on Facebook
RSS Feed
Follow us on Twitter

Copyright 2012. Designed by bytech
Privacy Policy | Terms of Use | XHTML | CSS

Figure 5.12: Example of placing an order for an item

This interface shows how an order for an item is placed. The client enters their shipping details for the item to be delivered and click on the order button. On the success of the order placing the results will be shown as per figure below.



Nongoma Online Solution
Let make your business run like clock work

Place an order

Item Name:	Trendy Tricket
Origin:	Art&Craft
Price:	$10.0
Quantity:	5
Shipping Address:	131 Ohio, USA
E-mail:	mrtshezi@gmail.com

Thank you for your order, a notification e-mail will be sent to mrtshezi@gmail.com

Services
- Service Number 1
- Service Number 2
- Service Number 3
- Service Number 4
- Service Number 5
- Service Number 6

Contact Information
1234 SomeStreet
Nongoma , SA 3886
Phone: 1(234) 567 8910
Fax: 1(234) 567 8910
E-mail:
nongomaonlinesolution@yahoo.com

Recent Posts
Nongoma .Post's e-commerce solution, Sell Online Store, is ideal for entrepreneurs who want to sell their products over the Internet.

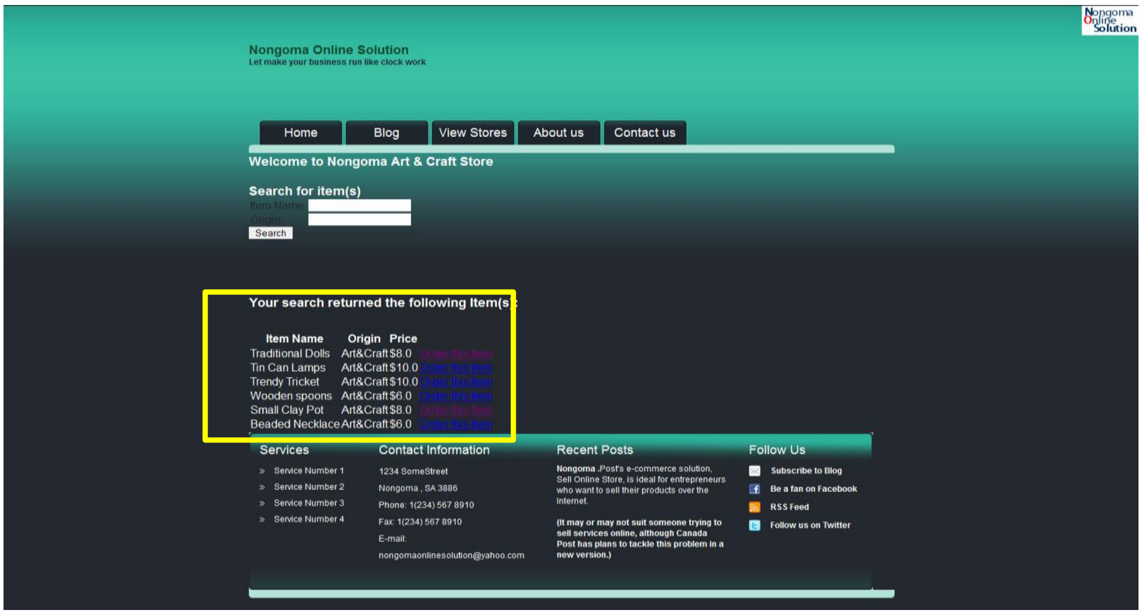
(It may or may not suit someone trying to sell services online, although Canada Post has plans to tackle this problem in a new version.)

Follow Us

Copyright 2012. Designed by bytech

Figure 5.13: Example of an order confirmation

This interface shows the confirmation for both an order and an e-mail notification for the order received. The client sees this notification for a successful order placement and awaits an e-mail notification containing details of when to expect to receive the order.



Welcome to Nongoma Art & Craft Store

Search for item(s)

Item Name:

Origin:

Your search returned the following Item(s):

Item Name	Origin	Price	
Traditional Dolls	Art&Craft	$8.0	Order this item
Tin Can Lamps	Art&Craft	$10.0	Order this item
Trendy Tricket	Art&Craft	$10.0	Order this item
Wooden spoons	Art&Craft	$6.0	Order this item
Small Clay Pot	Art&Craft	$8.0	Order this item
Beaded Necklace	Art&Craft	$6.0	Order this item

Services

» Service Number 1
» Service Number 2
» Service Number 3
» Service Number 4

Contact Information

1234 SomeStreet
Nongoma , SA 3886
Phone: 1(234) 567 8910
Fax: 1(234) 567 8910
E-mail:
nongomaonlinesolution@yahoo.com

Recent Posts

Nongoma . Post's e-commerce solution, Sell Online Store, is ideal for entrepreneurs who want to sell their products over the Internet.

(It may or may not suit someone trying to sell services online, although Canada Post has plans to tackle this problem in a new version.)

Follow Us

Subscribe to Blog
Be a fan on Facebook
RSS Feed
Follow us on Twitter

Figure 5.14: Example of the products catalogue

The figure above is an example of the items that are available in the store. These are the items that are currently available in the store catalogue.

5.6 Chapter Summary:

The chapter started by introducing the methodology applied, this was then followed by an in-depth view of the selected open-source solutions used as applicable to this project. A generic sequence diagram was presented to show sequence of any GUISET-based infrastructure that was then followed by a walk through the sample GUISET application the NOS, its user interface design showing an example of an order for arts.

Chapter 6: Conclusions and Future Work

6.1 Introduction:

The goal of this research work was to prototype the GUISET infrastructure using a combination of existing tools to support the e-business enablement of SMMEs to conduct their business activities online. The GUISET infrastructure is put together from existing open-source solutions and the NOS application has been prototyped to support service composition as components have to be assembled to achieve some business goal and also supporting dynamic service binding to enable late binding to the services defined by the assembled components.

This chapter begins by presenting, in Section 6.2, the summary in realisation of the GUISET infrastructure and NOS application which is supported by the infrastructure; the achievements of this research work are highlighted. The limitations and future directions of this work are given in section 6.3.

6.2 Conclusion:

There is a very high demand for infrastructures that enable conducting business activities online. This is especially true for SMMEs trading in rural areas with minimal budget to get exposure to the broader market brought about by the Internet. The GUISET infrastructure has been proposed as such an infrastructure to enable the usage of e-business for SMMEs in rural areas. The GUISET infrastructure is demonstrated through combining existing tools to address service composition and dynamic binding. The GUISET infrastructure is presented as an abstract architecture presented in Chapter 2, this is adopted from the on-going research on the GUISET concept. The architecture presented in Chapter 5 that takes a closer look at approach taken by GUISET in addressing the issues of service composition and dynamic service binding. The GUISET characteristics laid the foundation for the software development paradigm adopted. A comparison of different paradigms was conducted to select the candidate development paradigms for the development of an infrastructure that conforms to the GUISET based requirements. These GUISET requirements are specifically for the context of this work and these requirements were mostly met by SOC. This then restricted the tool selection to only the tools supporting SOC as the idea was they would make the implementation easier. Several works were explored to find tools that would be adopted for the infrastructure reported here. The process of exploring each tool was made easier as there was extensive documentation for the tools that were explored.

The GUISET infrastructure as it has been developed using the SCA specifications made the development a lot easier. The development came mostly through using some examples provided online, extending and modifying capabilities to meet the specific needs of GUISET. Apache Tuscany came packaged with several examples that one can learn from, the one watched closely for this work was that of an online store to get a clear view of how it addressed service composition; this was explored together with possibilities of finding an approach that would allow dynamic binding to service to achieve a flexible environment. Apache Tuscany is built based on the SCA specifications. Both CBSE and SOA focus reusing existing software components in their development approach. The choice of using SOA over CBSE was mostly informed by the support of GUISET requirements and in addition the tool support was deemed sufficient especially for the development of a dynamic environment. The SCA specifications based on the assembly model created an assembly of components based on some business goal and then used the OSGi runtime to ensure that the services used by these components are dynamically bound to and finally used Apache CXF to enable usage of services implemented in any technology.

6.3 Limitations and Future Work:

The market is overwhelmed with these open-source solutions aimed at achieving some business goal. Some open-source solutions are used independently and others are combined to work together, like the solution provided for the GUISET infrastructure which has seen the combination of existing open-source solutions. This work showed how the GUISET infrastructure can be put together supporting service composition and dynamic service binding by combining readily available solutions effectively. The limitations of this work entails not giving a fair judgement of how the infrastructure would actually perform in a real scenario as only the minimal functionality has been implemented. In the near future the intention is to build upon this work to make the GUISET infrastructure ready for usage by its member users. Some of the issues that will be addressed are how the GUISET infrastructure can support localisation of interfaces for the Nongoma community and how it can support the diverse language choices of its users. The evolutionary prototyping approach selected for this work will make the process of applying these changes to the prototype easier through further iterations of the prototype being developed. Compared to the requirements that were addressed in the reported work which was only concerned with demonstrating a typical GUISET configuration, the evaluation of the iterations developed based on the identified future work will require users to evaluate the prototype to determine if the developed iteration of the prototype has met the intended requirements. Further to that the intention is also to implement an infrastructure supporting other applications such as e-health, e-booking and other ideas that might come along. Since GUISET is realisable as a prototype other ideas that might be worth pursuing is finding ways of integrating existing works that have been done on the GUISET concept. Among other works that are worth mentioning is the work of Buthelezi et al, (2008), this work can provide a pricing model for the GUISET services, the work of Mathonsi (2011) can provide quality of service measure for selecting services to be composed to achieve a business goal and Mhlongo (2011) can provide for an authentication framework to ensure the GUISET infrastructure is secured. With the works I have just mentioned, which is among the few that has been done in the department, these are thought to be the closest in easy integration and are of vital importance in the success of the GUISET infrastructure.

6.4 Chapter Summary:

The chapter has presented the summary of the research work that has been conducted in realising the GUISET infrastructure and how the NOS application supported by the GUISET infrastructure validates the infrastructure. The user experience is presented to outline the process taken in the development of the infrastructure. The limitations and future directions of the work were discussed to show there is still further work to be done on GUISET as only the first prototype was presented here it mainly concentrated on some narrow requirements of GUISET.

References

Adigun, M.O., Emuoyibofarhe, O.J., Migiro, S.O. (2006). Challenges to Access and Opportunity to use SMME enabling Technologies in Africa, A presentation at 1st All Africa Technology Diffusion Conference, Johannesburg, June 12-14,

Adigun, M.O. (2007). E-Infrastructure Support for Rural Small and Micro Enterprises, SME: A Case Study of South Africa, World IT Forum 2007, 22-24 August 2007, http://www.witfor.org/2007/www.witfor2007.org/commission/building-the-infrastructure/GUISEWITFOR.pdf (Last accessed on 30 June 2012)

Adigun, M., Iyilade, J., Kabini, K. (2010). Agent-Based Infrastructure for Dynamic Composition of Grid Services. In N. Antonopoulos, G. Exarchakos, M. Li, & A. Liotta (Eds.), *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications*, (pp. 911-936). Hershey, PA: Information Science Reference. doi:10.4018/978-1-61520-686-5.ch039

Akoumianakis, D. (2008). Distributed Knowledge Management in Virtual Organizations: the 'Social' Experience Factory, 2008, The Electronic Journal of Knowledge Management, (Vol. 6, No. 1), pp. 13-32, http://www.aiai.ed.ac.uk/~jessicac/project/KMM/Reading%20List/16-DKM-Akoumianakis-2008.pdf (Last accessed on 29 August 2012)

Al-Jaroodi, J., Mohamed, N., Aziz J. (2010). Service Oriented Middleware: Trends and Challenges, 2010 Seventh International Conference on Information Technology, pp. 974-979

Avison, D., & Fitzgerald, G. (2006). *Information Systems Development: Methodologies, Techniques and Tools*, 4th Edition, Illustrated, McGraw-Hill Education, 2006

Blinco, K., Grisby, T., Laird, A., O'Neill, O., Srikanth, V., Smythe, C. (2009). Adoption of Service Oriented Architecture (SOA) for Enterprise Systems in Education: Recommended Practices: Original White Paper, 14 September 2009, http://www.imsproject.org/soa/soawpv1p0/imsSOAWhitePaper_v1p0.html (Last accessed on 30 December 2011)

Boochi, L., Fiadeiro, J.L., Lopes, A. (2008). Service-Oriented Modelling of Automotive Systems, 32nd Annual IEEE International Computer Software and Applications Conference, (COMPSAC '08), 28 July – 1 August 2008, pp. 1059-1064

Brada, P. (2008). Enhanced OSGi Bundle Updates to Prevent Runtime Exceptions, 34th Euromicro Conference Software Engineering and Advanced Applications, 2008, (SEAA '08), 3-5 September 2008, pp. 92-99

Breivold, H.P., Larsson, M. (2007). Component-Based and Service-Oriented Software Engineering: Key Concepts and Principles, 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, (2007), 28-31 August 2007, pp. 13-20

Bucchiarone, A., Gnesi, S. (2006). A Survey on Services Composition Languages and Models, International Workshop on Web Services Modelling and Testing, (WS-MaTe 2006), http://www.selab.isti.cnr.it/ws-mate/Bucchiarone_WS-MaTe.pdf (Last accessed on 30 December 2011)

Buthelezi, M.E., Adigun, M.O., Ekabua, O.O., Iyilade, S.S. (2008). Accounting, Pricing and Charging Service Models for a GUISET Grid-Based Service Provisioning Environment, Proceedings of the 2008 International Conference on E-Learning, E-Business, Enterprise Information Systems, and E-Government, (EEE 2008), 14-17 July 2008, pp. 350-355

Chappel, D.A. (2004). *Enterprise Service Bus: Theory in Practice*, O'Reilly Media

Chappell, D. (2007). Introducing SCA, July 2007, http://www.davidchappell.com/articles/introducing_sca.pdf (Last accessed on 14 October 2011)

Cheesman, J., Ntinolazos, G. (2004). The SOA Reference Model, CBDI Journal 2004, http://stratasoftware.com/downloads/CBDIWP-SOARMPart1.pdf (Last accessed on 15 August 2012)

Chiliya, N., Chikandiwa, C.K., Afolabi, B. (2011). Factors Affecting Small Micro Medium Enterprises' (SMMEs) Adoption of E-Commerce in the Eastern Cape Province of South Africa, International Journal of Business and Management, October 2011 (Vol.

6, No. 10), pp. 28-36, http://dx.doi.org/10/5539/ijbm.v6n10p28 (Last accessed on 07 July 2012)

Chu, Q., Shen, Y., Jiang, Z. (2009). A Transaction Middleware Model for SCA Programming, International Workshop on Education Technology and Computer Science, 2009, (ETCS '09), 7-8 March 2009, pp.568-571

Cloete, E., Courtney, S., Fintz, J. (2002). Small Businesses' Acceptance and Adoption of e-commerce in the Western-Cape Province of South Africa, Electronic Journal on Information Systems in Developing Countries, 2002, EJISDC, (Vol. 10 No. 4), pp. 1-13

COFISA Project Plan, (2008). Rural ICT Mapping Exercise in the Eastern Cape and Western Cape, Step 3.3 – Rural Development and ICTs, Final Version: May 2008, http://unpan1.un.org/intradoc/groups/public/documents/cpsi/unpan038413.pdf (Last Accessed on 31 August 2012)

Crnkovic, I., Larsson, M. (2002). *Building Reliable Component-Based Software Systems*, Artech House.2002, Norwood

Dahiya, D. (2010). Enterprise Systems Development: Impact of Various Software Development Methodologies, 2nd International On Software Engineering and Data Mining (SEDM), 23-25 June 2010, pp. 177-122

Dalvit, L., Thinyane, M., Muyingi, H., Terzoli, A. (2007). The Deployment of an e-Commerce Platform and Related Projects in a Rural Area in South Africa, International Journal of Computing and ICT Research, 2007, (Vol. 1, No. 1), pp. 9-18, www.ijcir.org/downloads/article1.pdf (Last accessed on 29 August 2012)

Davis, J. (2009) *Open Source SOA*, Manning Publications Co., 2009

DTI, The. (2005). Integrated Strategy on the Promotion of Entrepreneurship and Small Enterprises, The Department of Trade and Industry: Republic of South Africa, 2005 http://www.thedti.gov.za/sme_development/docs/strategy.pdf (Last Accessed on 11 August 2012)

Dyakalashe, S. (2009). Cultural and Linguistic Localization of the Virtual Shop-Owner Interface of E-Commerce Platforms for Rural Development, University of Fort Hare, November 2009, Eastern Cape, South Africa (unpublished)

Eichberg, M. (2005). Component-Based Software Development with Aspect Oriented Programming, http://www.jot.fm/issues/issue_2005_04/article3/ (Last Accessed: 15 March 2012)

Ekabua, O.O., (2009). Change Impact Analysis Model-Based Framework for Service Provisioning in a Grid Environment, University of Zululand, February 2009, KwaZulu-Natal, South Africa (unpublished)

Ekabua, O.O., Adigun, M.O. (2010). GUISET LogOn: Design and Implementation of GUISET-driven Authorization Framework, 1st International Conference on Cloud Computing, GRIDs and Virtualization, 21-26 November 2010, pp. 1-7

Etienne, J.P., Bouzefrane, S. (2006). Applying the CBSE paradigm on Real-time Systems, 2006 International Workshop on Factory Communications Systems, vol., no., pp. 368-373

Farrel, A. (2007). Selecting a Software Development Methodology based on Organization Characteristics, Athabasca University, October, Canada (unpublished)

Ferguson, C., Finn, F., Hall, J., Pinnuck, M. (2010). Speculation and e-commerce: The long and the short of IT, International Journal of Accounting Information Systems 11, ScienceDirect, pp. 79-104

Foster, I., Kesselman, C., Tuecke, S. (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organisations, International Journal of High Performance Computing Applications, Fall 2001, (Vol. 15, No. 3), pp. 200-222

Gomez, J.M., Alor-Hernandez, G., Posada-Gomez, Rivera I., Mencke, M., Chamizo, J., Sanchez, F.G., Toma, I. (2008). An approach for Component-based Software Composition, Electronics, Robotics and Automotive Mechanics Conference, (CERMA '08), Vol., No., 30 September – 03 October 2008, pp. 195-200

Guinard, D., Trifa, V., Wilde, E. (2010) A Resource Oriented Architecture for the Web of Things, Internet of Things, IEEE, 29 Nov – 1 Dec 2010, pp. 1-8

Guner, S. (2005). Architectural Approaches, Concepts and Methodologies of Service Oriented Architecture, Software System Institute, Technical University Hamburg Germany, August 2005, (Unpublished)

Gunestas, M. (2005) A Study on Component Based Software Engineering, Atilim University, Ankara, Turkey, January 2005 (unplublished)

Guo, X., Shen, J., Yin, Z. (2010). On Software Development Based on SOA and ROA, 2010 Chinese Control and Decision Conference, Vol., No.,, 26-28 May 2010, pp. 1032-1035

Haapasalo, T. (2007). Using Open-Source Solutions in Agile Software Development, Helsinki University of Technology, January 2007, Finland (Unplublished)

Herselman, M.E. (2003). ICT in Rural Areas in South Africa: Various Case Studies, Proceedings of Informing Science, InSITE – "Where Parallels Intersect", http://proceedings.informingscience.org/IS2003Proceedings/docs/120Herse.pdf (Last accessed on 11 August 2012)

Hesari, S., Mashayekhi, H., Ramsin, R. (2010). Towards a General Framework for Evaluating Software Development Methodologies, 34th Annual IEEE Computer Software and Applications Conference, 19-23 July 2010, 208-217

Houspanossian, A. (2006). Enhancing a BPEL4WS Engine Supporting the Execution of Flexible WS-flows According to the ReFFlow Model, Universidad Nacional del Centro de la Provincia de Buenos Aires, March, 2006, Tandil, Argentina (unpublished).

Igbaria, M., Zinatelli, N., Cragg, P., Cavaye, A.L.M. (1997). Personal Computing acceptance factors in Small Firms: A Structural Equation Model, MIS Quarterly September, 1997, (Vol. 21 No. 3), pp. 279-305, Minneapolis, http://www.jstor.org/stable/pdfplus/249498.pdf (Last accessed on 30 July 2012)

Internet World Stats. (2011). http://www.internetworldstats.com/stats.htm (Last accessed on 30 October 2011)

Ishikawa, H., Ogata, Y., Adachi, K., Nakajima, T. (2004). Building Smart Appliance Integration Middleware on the OSGi Framework, Proceedings of the 7th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'04), Vienna, 12-14 May 2004

Iyilade, J.S., Kabini, K., Adigun, M. (2009). MINDS: A Middleware Infrastructure for Distributed Services Provisioning, 6th International Conference on Information Technology: New Generations, 27-29 April 2009, pp. 1018-1023

Iyilade, J.S. (2010). Grid-Based Utility Middleware Infrastructure for Distributed Services Provisioning, University of Zululand, April 2010, KwaZulu-Natal, South Africa (unpublished)

Jarvensivu, J., Kosola, M., Kuusipalo, M., Reijula, P., Mikkonen, T. (2006). Developing an Open Source Integrated Development Environment for a Mobile Device, International Conference on Software Engineering Advances, October 2006, pp. 55

Joseph, J., Ernest, M., Fellenstein, C. (2004). Evolution of grid computing architecture and grid adoption models, IBM Systems Journal, October 2004 (Vol. 43 No. 4), pp. 624-645

Ketfi, A., Belkhatir, N. (2005). Model-Driven Framework for Dynamic Deployment and Reconfiguration of Component-Based Software Systems, In Proceedings of the 2005 symposia on Metainformatics, (MIS '05) ACM, New York, Article 8

Kiczales, G., Mezini, M. (2005). Aspect-Oriented Programming and Modular Reasoning, In Proceedings of the 27th International Conference on Software Engineering, 2005, (ICSE'05), 15-21 May 2005, pp. 49-58

Kitchenham, B.A., Pfleeger, S.L., Hoaglin, D.C., Rosenberg, J. (2002). Preliminary Guidelines for Empirical Research in Software Engineering, IEEE Transactions on Software Engineering, August 2002, (Vol. 28, No. 8), pp. 721 - 734

Koskela, M., Rahikainen, M., Wan, T. (2007). Software development methods: SOA vs. CBD, OO and AOP, Helsinki University of Technology, http://www1.soberit.hut.fi/T-86/T-86.5165/2007/final_koskela_rahikainen_wan.pdf (Last accessed on 30 June 2012)

Kozuch, M., Mahadev Satyanarayanan Bressoud, T., Helfrich, C., Sinnamohideen, S. (2004). Seamless mobile computing on fixed infrastructure, Computer, (Vol.37, No.7), July 2004, pp. 65-72

Kryvinska, N., Strauss, C., Collini-Nocker, B., Zinterhof, P. (2010) A Scenario of Service-oriented Principles Adaptation to the Telecom Providers Service Delivery Platform, Fifth International Conference on Software Engineering Advances, 22-27 August 2010, pp. 265-271

Kshetri, N. (2007). Barriers to e-commerce and competitive business models in developing countries: A case study, Electronic Commerce Research and Applications 6, 2007, (ScienceDirect), pp. 443-452

Laws, S., Combellack, M., Feng, R., Mahbod, H., Nash, S. (2011). *Tuscany SCA in Action*, Manning Publications Co, Greenwich

Li, W., Zhang, Y., Jin, J. (2009). Research of The Service Design Approach Based on SCA_OSGi, IITA International Conference on Services Science, Management and Engineering, 2009, (SSME '09), 11-12 July 2009, pp. 392-395

Li, Z., Parashar, M. (2006). An Infrastructure for Dynamic Composition of Grid Services, 7th IEEE/ACM International Conference on Grid Computing, 28-29 September 2006, pp. 315-316

Liping, Y., Kai, S. (2010). Design and Realization of Image Processing System Based on Embedded Platform, International Forum on Information Technology and Applications, (Vol. 2, no.), 16-18 July 2010, pp. 446-449

Llorente, I.M., Montero, R.S., Huedo, E., Leal, K. (2006). A Grid Infrastructure for Utility Computing, 26-28 June 2006, pp. 163-168

Maarouf, M.Y., Chung, S.M. (2008). XML Integrated Environment for Service-Oriented Data Management, 20th IEEE International Conference on Tools with Artificial Intelligence, 3-5 November 2008, pp. 361-368

Maican, C.I. (2009). Integrated University Management System Based on Open Source Tools, Fourth International Conference on Internet and Web Applications and Services, 24-28 May 2009, pp. 626-631

Maigre, R. (2010). Survey of the Tools for Automating Service Composition, IEEE International Conference on Web Services, vol., no., 5-10 July 2010, pp. 628-629

Management Services, Computers Fail to Click with Small Businesses, Enfield, (Vol. 41, No. 4) Institute of Management Services, 1997, HighBeam Research: http://www.highbeam.com/doc/1P3-14526989.htm (Last accessed on 30 September 2012)

Marino, J., Rowley. M. (2009). *Understanding SCA (Service Component Architecture)*, Addison Wesley, 2009, Indianapolis

Masek, K., Hnetynka, P., Bures, T. (2009). Bridging the component-based and service-oriented worlds, Conference on Software Engineering and Advanced Applications, 35th Euromicro, 27-29 August 2009, pp. 47-54

Maswera, T., Dawson, R., Edwards, J. (2008). E-commerce adoption of travel and tourism organisations in South Africa, Kenya, Zimbabwe and Uganda, Telematics and Informatics 25, ScienceDirect, 2008, pp. 187-200

Mathonsi, E.L. (2011). Development of a Web Service Client Framework for GUISET, University of Zululand, July 2011, KwaZulu-Natal, South Africa (unpublished)

Medvidovic, N., Mikic-Rakic, M., Mehta, N.R., Malek, S. (2003). Software architectural support for handheld computing, Computer, September 2003, (Vol.36, No.9), pp. 66-73

Migiro, S.O., Adigun, M.O. (2005). ICTs, e-Commerce and Rural Development: The case of arts and crafts SMEs in rural KwaZulu-Natal, Commonwealth Youth and Development, 2005, (Vol. 3, Issue 2), pp. 65-83

Minoli, D. (2008). *Enterprise Architecture A to Z: Frameworks, Business Process Modelling, SOA, and Infrastructure Technology*, CRC Press, 2008

Modimogale, L., Kroeze, J.H. (2011). The Role of ICT within Small and Medium Enterprises in Gauteng, Communications of IBIMA, (Vol. 2011), ArticleID: 369288, http://www.ibimapublishing.com/journals/CIBIMA/2011/369288/369288.pdf (Last Accessed on 01 July 2012)

Mulik, S. (2009). Using Enterprise Service Bus (ESB) for connecting Corporate Functions and Shared Services with Business Divisions in a large Enterprise, 2009 IEEE Asia-Pacific Services Computing Conference, 7-11 December 2009, pp. 430-434

Murphy, G.C., Kersten, M., Findlater, L. (2006). How Are Java Software Developers Using the Eclipse IDE?, IEEE Software, July-August. 2006, (Vol. 23, No. 4), pp. 76-83

Nadiminti, K., Buyya, R. (2005). Enterprise Grid computing: State-of-the-Art, October 2005, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, http://www.cloudbus.org/reports/EOSJArticleTR05.pdf (Last accessed on 30 June 2012)

Najdawi, A. (2009). SOA and Web Services for Leveraging Inter-Organizational Integration in Travel and Tourism Industry, 2009 World Conference on Services – I, 6-10 July 2009, pp. 151-154

Nakagawa, E.Y., Machado de Sousa E.P., Mura, K., Andery, G., Morelli, L.B., Maldonado, J.C. (2008). Software Architecture Relevance in Open Source Software Evolution: A Case Study, 32nd Annual IEEE International Computer Software and Applications Conference, COMPSAC, 28 July – 1 August 2008, pp. 1234-1239

Nigul, L., Kontogiannis, K., Brealey, C. (2009). The SOA programming model: challenges in a services oriented world, 2009 Conference of the Center for Advanced Studies on Collaborative Research (CASCON '09), pp. 341-342.

Ning, W., Liming, L., Yanzhang, W., Yi-bing, W., Jing, W. (2008). Research on the Web Information System Development Platform Based on MVC Design Pattern, IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 9-12 December 2008, pp. 203-206

Njeje, S.G. (2008). Implementing a robust, cost effective, e-commerce platform for a disadvantaged community of the Eastern Cape, South Africa, University of Fort Hare, March 2008, Eastern Cape, South Africa (unpublished)

Nyholm L, Liz in Africa, http://lizinafrica.posterous.com/umkhosi-womhlanga-royal-reed-dance (Last accessed: 30 December 2011)

Obrenovic, Z., Gasevic, D. (2007). Open Source Software: All You Do Is Put It Together, IEEE Software, September – October 2007, (Vol 24, No. 5), pp. 86-95

Ommering, R., Linden, F., Kramer. J., Magee, J. (2000). The Koala Component Model for Consumer Electronics Software, Computer, Mar 2000, (Vol. 33, No.3), pp. 78-85

Oreku, G.S. (2010). Open Source Software Application and their Impacts on SMEs: an Action for Building an OSS Community environment in Tanzania, IST-Africa 2010, vol, no, 19-21 May 2010, pp. 1-8,

OSGi Alliance, OSGi Alliance Core Specifications Version 4.3, http://www.osgi.org/download/r4v43/r4.core.pdf (last accessed on 14 October 2011).

Overdick, H. (2007). The Resource-Oriented Architecture, Proceedings of the 2007 IEEE Congress on Services, 9-13 July 2007, pp. 340-347

Papazoglou, P.M., Georgakopoulos, D. (2003). Introduction: Service-oriented Computing, ACM Communications, ACM 46, 10 October 2003, pp. 23-28

Papazoglou, P.M., Traverso, P., Dustdar, S., Leymann, F., Kramer, B.J. (2006). Service-Oriented Computing Research Roadmap, http://infolab.uvt.nl/pub/papazogloump-2006-96.pdf (Last accessed on 30 December 2011)

Pastrana, A., Lopez, E.S. (2009). Possibilities of Open Source Software in Developing Local Small Business, International Conference on Intelligent Networking and Collaborative Systems, 4-6 November 2009, pp. 413-416

Pieber, A., Spoerk, J. (2008). A Comparative Analysis of State-of-the-Art Component Frameworks for the Java Programming Language, 25 September 2008, http://cocoon.ifs.tuwien.ac.at/lehre/praktikumsarbeiten/2008_pieber_component_frameworks.pdf (Last accessed on 30 June 2012)

Pollard, C.E., Hayne, S.C. (1998). The Changing Faces of Information Systems Issues in Small Firms, International Small Business Journal, April 1998 (Vol. 16, No. 3), pp. 70-87, http://isb.sagepub.com/content/16/3/70.abstract (last accessed on 31 August 2012)

Ramollari, E., Dranidis, D., Simons, A.J.H. (2007). A Survey of Service Oriented Development Methodologies, 2nd European Young Researchers Workshop on Service Oriented Computing, University of Leicester, UK, 11-12 June 2007

Ruiz, J.L., Duenas, J.C., Cuadrado, F. (2008). A Service Component Deployment Architecture for e-Banking, International IEEE Workshop on Service Oriented Architectures in Converging Networked Environments, (SOCNE 08), 25-28 March 2008, pp. 1369-1374

Savva, A., Suzuki, T., Kishimoto, H. (2004). Business Grid Computing Project Activities, Fijutsi Science Tech Journal, Dec. 2004, (Vol. 40, No. 2), pp. 250-262, http://www.fujitsu.com/downloads/MAG/vol40-2/paper09.pdf (Last Accessed on 25 March 2012)

Seinturier, L., Merle, P., Fournier, D., Dolet, N. (2009). Reconfigurable SCA Applications with the FraSCAti Platform, 6th IEEE International Conference on Service Computing, pp. 268-275

Seinturier, L., Merle, P., Rouvoy, R., Romero D., Schiavoni, V., Stefani, J. (2012). A Component-Based Middleware Platform for Reconfigurable Service-Oriented Architectures, Software: Practice and Experience, 2012, (Vol. 42, No.5), pp. 559-583

Sibiya, M.G., Jembere, E., Xulu, S.S., Adigun, M.O. (2008). A Web Services based e-Commerce Business Model for Resource Constrained SMMEs, Proceedings of the 2008 SATNAC Conference

Sibiya, M.G. (2010). Context-aware Service Discovery for Dynamic Grid Environments, University of Zululand, October 2010, KwaZulu-Natal, South Africa (unpublished)

Sibiya, S.S. (2009). Dynamic Service Recovery in a Grid Environment, University of Zululand, October 2009, KwaZulu-Natal, South Africa (unpublished)

Sirbi, K., Kulkarni, P.J. (2010). Modularization of Enterprise Application Security Through Spring AOP, International Journal of Computer Science and Communication, July-December 2010, (Vol. 1, No. 2), pp. 227-231,

Sommerville, I. (2001), *Software Engineering*, 6th Edition, Addision-Wesley

SQA FIVT 34. Prototyping, http://www.sqa.org.uk/e-learning/IMAuthoring01CD/page_04.htm#Proto (Last accessed on 18 July 2011).

Srinivasan, L., Treadwell, J. (2005). An Overview of Service-oriented Architecture, Web Services and Grid Computing, HP Software Global Business Unit, White Paper, 3 November 2005, (Last Accessed : 10 November 2011)

Sun, C. (2011). China's E-Commerce Becomes Reality, 2011 International Conference on E-Business and E-Government, (ICEE), 6-8 May 2011, pp. 1-4

Sun Microsystems, Inc., Guide to Using Open-Source Software to Develop Web Applications, Open Web Application Platform. White Paper, April 2009

Tan, W., Tian, Z., Rao, F., Wang, L., Fang, R. (2006). Process Guided Service Composition in Building SOA Solutions: A Data Driven Approach, International Conference on Web Services, Vol., no., 18-22 September 2006, pp. 558-568

Tosic, V., Mennie, D., Pagurek, B. (2001). On Dynamic Service Composition and Its Applicability to E-business Software Systems, Workshop on Object-Oriented Business Solutions, Budapest Hungary, 18 June 2001

Wang, J. (2006). Web Service Discovery and Invocation Assistant Service, http://technology.asu.edu/files/documents/tradeshow/Dec06/ProjectReport_WSDIA_JunWang.pdf (Last accessed on 28 October 2010)

Wang, Y., Guo, C., Song, L. (2009). Architecture of E-Commerce Systems Based on J2EE and MVC Pattern, International Conference on Management of e-Commerce and e-Government, 16-19 September 2009, pp. 284-287

Warden, S.C., Motjolopane, I.M. (2007). E-commerce Adoption Factors for SMMEs: Supporting Cases from South Africa, Managing Worldwide Operations and Communications with Information Technology, pp. 701-709, http://www.irma-international.org/viewtitle/33168/ (Last accessed on 29 August 2012)

Weaver, R. (2009). The Business Value of the Service Component Architecture (SCA) and Service Data Objects (SDO), November 2009

Wesner S, Jahnert J.M. (2006). Mobile Collaborative Business Grids – A short overview of the Akogrimo Project, pp 1-6, http://www.mobilegrids.org/download/White_Papers_and_Publications/Akogrimo_White Paper_Overview.pdf; (Last accessed: 12 February 2014)

Wu, C., Liao C., Fu, L. (2007), Service-Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, March 2007 (Vol. 37, No. 2), pp. 193-205

Xiong-Yi, L. (2009). Research and Application of SOA in B2B Electronic Commerce, International Conference on Computer Technology and Development, IEEE, 13-15 November 2009

Yan, J., Jian, W., Zheng, S., Wu, C., Pan, H. (2009). Research on application of Web based ESB in School Common Data Platform, Computer Science and Education, 4th International Conference on Computer Science and Education, 25-28 July 2009, Nanning

Yung-Wei, K., Chia-Feng, L., Kuei-An, Y., Shyan-Ming, Y. (2011). A Cross-Platform Runtime Environment for Mobile Widget-Based Application, International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 10-12 October 2011, pp. 68-71

Zhang, B., Xu, T., Wang, W., Jia, X. (2011). Research and implementation of cross-platform development of mobile widget, 3rd International Conference on Communication Software and Networks, 27-29 May 2011, pp. 146-150

Zhang, L., Li, H., Lam, H. (2004). Toward a business process grid for utility computing, IT Professional, September - October 2004, (Vol. 6, Issue: 5), pp. 62-64

Zhengyan, D. (2011). Aspect Oriented Programming Technology And The Strategy Of Its Implementation, 2011 International Conference on Intelligence Science and Information Engineering, ISIE, 20-21 August 2011, pp. 457-460

Zhou, C., Nu,i L. (2010). Research on Component Based Online Shopping System Design, International Conference on Computational Aspects of Social Networks, vol., no., 26 – 28 September 2010, pp. 133-136

Zululand Tourism. (2009). Zululand Tourism, http://www.zululandtourism.org.za/Tourism/details/nongoma.aspx?ShowAspMap=nongoma (Last accessed: 30 December 2011)

Prototyping the GUISET Architecture using the Nongoma Online Stores

Siyabonga Sifiso Cebekhulu

(20034365)

A dissertation submitted in fulfilment of the requirements for the degree of

Master of Science in Computer Science

Department of Computer Science, Faculty of Science and Agriculture

University of Zululand

Supervisor: Professor M.O. Adigun

2014