Dynamic Composition of Portal Interface for GUISET Services

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

Master of Science (Computer Science)

by

Siyasanga Amanda Sihawu

200906057

Department of Computer Science

Faculty of Science and Agriculture, University of Zululand

Supervisor: Prof M.O. Adigun

Co-Supervisor: Prof S.S. Xulu

2014

DECLARATION

I, Sihawu Siyasanga Amanda declare that this dissertation represents my work and it has not been submitted in any form for another degree or diploma at any University or any other institution of tertiary education. All information taken from published and unpublished works of others has been well acknowledged in the text and a list of respective references is given.

DEDICATION

I dedicate this work to both Ganamfana and Sihawu families for believing in me and for the support throughout the course of this work.

ACKNOWLEDGEMENTS

I would like to thank the Lord for the strength that he has given me throughout this work. Many thanks to my supervisor Professor M.O. Adigun and co-supervisor Professor S.S. Xulu for the support and guidance. A special thank you to Edgar Jembere for the support, patience and making this work realizable. Many thanks to Dr. John. B Oladosu, Bethel Mutanga, George Sibiya, Pragasen Mudali and fellow researchers at the centre of excellence for their contributions.

A special gratitude to my family and to mama, Nontuthuzelo C. Ganamfana if it wasn’t for you I wouldn’t be where I am.

Contents

LIST OF TABLES

TABLE OF FIGURES

Abstract

The increased usage of technology requires the user interface to be user friendly and easy to understand. In portals, the user interface is one of the critical components. Portal is a medium that Small Medium and Macro Enterprises (SMME) can use to advertise, grow exposure of their business and interact with their customers and community at large. This requires the portal interface to adapt to runtime changes because of the rapid update of services.

This research work addressed the issue of analysis of user request in order to have a clear understanding of the request. This is achievable by taking into consideration the user history, preference and profile. This work is realizable through the use of the design of architectural model, the Dynamic User Interface (DUI) model which analyzes the user request and also aggregate different components and presents them as a single unit to the user. The DUI model was prototyped via Grid-based Utility Infrastructure for SMME-enabled Technology (GUISET) user interface use case. Experiments were conducted to compare DUI portal and Jetspeed-2 to test usability subjects. Usability was evaluated under four experiment variations: customization, adaptation after query, user satisfaction with response and portal performance. For each experiment, data was generated and graphically presented for analysis. The results obtained show scalability of the portal interface with regards to the increased number of users requesting for the same services and the increased number of portlets. The user satisfaction with the interface adaptation on the fly after the query and content is also shown on the results that were obtained after the experiments.

CHAPTER ONE

INTRODUCTION

1.1 Introductory Background

Web sites allow organisations to advertise not only themselves but also the services they provide. Unfortunately, in traditional web sites, clients are forced to go to different web sites in order to have a complete service and this may be time consuming. At times some users would give up before they accomplish what they wanted to do. This results in organizations being concerned about presentation of services to clients. To address this problem, organizations needed a site that can unify their services for their users. Web portals came as a result of this need. A web portal may be described as a web site that provides a gateway to other websites. Portals are used as gateways for a variety of services from different organizations. The usage of portals allows diverse services to be presented to clients from a single source.

Portal type can be either vertical or horizontal (Liu, Du, and Tsai, 2009; Schiefer and Kreuder, 2001). Vertical portals deal with one service domain, for example a portal about wines only deals with wines and nothing else. Horizontal portals have a range of different services, for example *yahoo.com* offers a broad range of services. Users that are looking for a specific item might not find it easily when they use a horizontal portal compared to using a vertical portal. In most cases, horizontal portals are information-based while vertical portals are transaction-based. Business to Consumer, Business to Business, and Business to Employee portals are vertical portals based on an assortment of goals and services which they offer (Credle, Coury, and Stimpfle, 2006). A Business to Consumer portal (B2C) supports customers of a specific company. A Business to Business (B2B) portal is about business to business relationships. Lastly a Business to Employee (B2E) is an intranet- based portal that provides employees with company resources.

Access to services via portal is through the portal's user interface .A portal's user interface is composed either at design time or at runtime. In a user interface that is defined at design time, the same sets of services are presented to each and every user. It is therefore possible that, in some instances, these static interfaces might not meet with every user needs. A user might need a service that was not defined at design time. A dynamic user interface allows users to define the services they want at runtime. When a user queries for an application at runtime, dynamic service composition takes place automatically. This may involve adaptation of running components and their functionalities through the addition or removal of service components.

Context influences the discovery of relevant services prior to user requests. In previous years, in discovery of services, most scholars considered user profile to help narrow down the discovery of services (Gauch et al, 2007; Kostkova, Diallo, and Jawaheer, 2008; Liu, Salem, and Rauterberg, 2009). This was not considering the current context of the user. Context may be divided into three categories according to Schilit, Adams and Want (1994):

a) *Computing context:* This is context information related to context aware system aspects. It entails system context that deals with network traffic, bandwidth and Quality of service etc.

b) *User context:* This is context information related to the service requestor. Personal context include health, mood, activity etc. Social contexts consist of social relationship and people around, etc.

c) *Physical context* is contextual information in related to the physical part of the Context Aware (CA) system. Examples of physical contexts are location and time, examples of environmental contexts are weather, light and examples of information contexts are sport scores, stock quotes, etc.

The awareness of the requestor context during the process of service discovery will help narrow down the matching of services and present more relevant services to the user interface. Context is dependent on the environment of the entity. An entity can be a person, location or an object. In Dey (2001), definition of context is focused on both the user and the application. Context is defined as “information that can be used to characterize the situation of an entity” (Dey,2001). Context information is the information about the context and its environment. This information is used for retrieving relevant services to the user‘s current state.

In the research conducted by Goncalves da Silva, Ferreira Pires, and van Sinderen (2009), an attempt was made to enhance service discovery based on the type of user. Their work’s main focus is the service composition on the fly, the interface and user context is not given much attention. The focus of this work is on the user context. It takes user context as a part of the content relevant to what the user has requested. Users should be able to specify the services even at runtime and get the services consistent with their context. The user interface should adapt to the user’s context.

This work uses the Grid based Utility Infrastructure for Small Medium and Macro Enterprises (SMME) Enabling Technology (GUISET) as a platform. GUISET is a research project based on the concept of Services (Service Oriented Architecture, web Services and Grid computing), Utility Computing and e-commerce (Adigun, Emuoyibofarhe, & Migiro, 2006). The main idea of GUISET is to provide an e-infrastructure which would enable SMMEs to pool their resources and expertise together for sharing and collaboration among themselves and their partners (Munćan, 2009). GUISET provides utility computing and service provisioning infrastructure and a dynamic portal interface will act as a presentation layer for GUISET. SMMEs as consumers will then utilize available services by interacting with the portal interface.

Portals in general have some common features; for example single access point, internet tools, collaboration tools etc (Munćan, 2009). GUISET portal interface will adopt those features and enhance them in a convenient manner for the SMME (Credle, Coury & Stimpfle, 2006):

a) The portal interface will allow customization.

b) There will be a business community, having discussion boards, community news and sharing of ideas among co-operating business groups.

c) The portal will support personalization of interface and services, when users have log in, the portal interface will be personalized to suit their profiles.

d) Adaptive interface to contextual changes on the fly (He & Yen, 2007; McKinley, Sadjadi, Kasten & Cheng, 2004; Savidis & Stephanidis, 2010; Yau & Karim, 2004; Zhao, Ninomiya, Anma & Okamoto, 2008).

1.2 Research Problem

Most e-commerce portals are built using existing tools and frameworks. These frameworks determine how the user interfaces are going to be integrated as one. The portal interface presents services that are available. Users interact with the portal through its interface; service requisition is part of their interaction (Hennoste *et al*., 2008; Pietschmann, Voigt, & Meissner, 2009; Singh & Wesson, 2006; Stephanidis, 2001).

Getting personalised relevant services becomes a problem in most portals. Personalized services are mostly determined by user history and user preferences. The history and preference narrow down the service discovery process and more relevant service are retrieved. There are other factors that influence the discovering of relevant services which include context.

The knowledge of the user's current environment influences the type of services to be presented. There are three different types of context: computational context, user context and physical context (Schilit, Adams & Want, 1994). This work seeks to analyze the user request so that it would be very clear for the discovery engine to select relevant services. The user interface should adapt to the changes that will be introduced at runtime.

In Patel and Chaudhary (2009), it is shown that one can get relevant results in context aware service discovery. The work of Patel and Chaudhary (2009) focused on service discovery algorithm which is based on rule engine. In this work, a model that will analyze the user request, checking its profile, history and preferences is proposed. The user interface should adapt to the changes at runtime. This model is concerned with user context and how it is used to select relevant content at runtime.

This should improve the user satisfaction, since static interface shows each and every user the same interfaces.

1.3 Rationale of the study

This research aims to enhance the Grid based Utility Infrastructure for SMME Enabling Technology (GUISET) proposed by Adigun, Emuoyibofarhe, and Migiro, (2006) to achieve user satisfaction by composing user interface on the fly and get relevant content. GUISET is an infrastructure aimed at supporting business processes for Small, Medium and Macro Enterprises (SMME) through shared Grid services. This dynamic portal interface will act as a presentation layer for GUISET. This will help the SMME's services to be discovered more easily because the user request will be analyzed in a way that will promote the relevance of services to be discovered. Portal users will benefit by getting relevant services that will be discovered at runtime.

1.4 Research Questions

This research work answers the following questions:

1. How can a portal interface adapt to user context?
2. How can context information enhance service request?

1.5 Goal and Objectives

1.5.1 Goal

The goal of this research is to propose dynamic portal user interface architectural model for GUISET services.

1.5.2 Objectives

In fulfilling the goal of this research, the following list of objectives was achieved:

a) Formulate a model for dynamic interface generation.

b) Prototype the model.

c) Evaluate the model using appropriate reliability metrics.

1.6 Research Methodology

The methods employed in this study are as follows:

1.6.1 Literature survey

The literature survey activity involved, firstly, a survey of interface composition and evaluation of the existing techniques used to compose interface. A survey of context in portal systems was also carried out. Secondly, portal architectures were investigated by analyzing existing architectures and identifying the criteria used when the architectures were built. Thirdly, tools and software used were also studied. After reviewing existing relevant literature, a case study and design criteria were formulated.

1.6.2 Design

Based on the case study and design criteria, a model was formulated. This model is aimed at addressing the issue of analysing the user request in order to get relevant content at run time. This is done by introducing a query analyzer that will help in the analysis of a user request.

1.6.3 Experiment

The formulated model was prototyped. NetBeans was used as an environment suitable for the proposed model. Liferay as open source framework was used and Java was used as the programming language. A dynamic portal interface was also built.

1.6.4 Data Collection and Analysis

The prototyped model was evaluated for (a) customization, (b) adaptation, (c) user satisfaction and (d) portal performance as a validation of our proposed solution. Questionnaires were used as a data collection instrument. Users were placed in three categories, Novice, Intermittent and Experts. Each experimental result is presented graphically for analysis.

1.7 Organization of the Dissertation

The rest of the dissertation is organized as follows:

Chapter Two presents the background concepts of this research work. It also lays a foundation for the proposed model. In Chapter Three, there is literature review. This chapter starts with an introduction on context in portal systems. Existing work and outlining the challenges associated with user interface generation are discussed. Chapter Four presents the description of the model development. This chapter begins with design requirements and a solution approach to solving the problem. The Dynamic User Interface Model is then presented in details. The implementation of the model is also discussed. A summary of this chapter is then presented. In Chapter Five we perform experiments, analysis, and performance evaluation of the results. Chapter Six concludes the dissertation and recommendations for the future work are also presented.

CHAPTER TWO

BACKGROUND

2.1 Introduction

In everyday use, people are manipulating portals to get services. They interact with a portal through its interface. This interface represents available services. In some instances the user can influence which services to be presented. This result into expectations of the user interface meeting user request. The user interface has to adapt to the user request. When services are discovered as part of the processes in order to present a service to a user, there are lot of factors that should be considered; including the context of the user. By including the current context of the user this will increase chances of getting relevant content. This would enhance user satisfaction because the user's desires would have been met. The user interface should also be scalable with regards to the number of portlets presented and the time taken to retrieve these services. In chapter one, it was indicated that the goal of this research is to build a dynamic portal interface for GUISET services. In this chapter we introduce background concepts which are fundamental in our research. Section 2.2 briefly discusses portals. This is followed by section 2.3 where there is a discussion of User Interface. Section 2.4 covers GUISET and Service Oriented Architecture. Section 2.5 briefly discusses User Interface adaptation. Finally, a summary of this chapter is given in section 2.6.

2.2 Portals

Most organisations deploy portals as a way of sharing resources. A portal is a web based application that acts as a gateway between users and a range of different high level services (Baker, Ong, Allan & Wang, 2004).There are two categories of portals namely horizontal and vertical portals (Liu, Du & Tsai, 2009). Vertical portals provide access to a variety of information and services from a particular area of interest. Horizontal portals provide access to a variety of information and services that are from different areas. In this work the focus is on horizontal portals. Portals have gained a lot of popularity due to ease of development, flexibility, customization of interface and pluggable architecture. The first generation of portals were designed with monolithic software with tightly coupled components and since then they have evolved. Portals provide single sign on, aggregation and personalization features. These portal features play a role in user loyalty because a user can be able to log in once and access service from different organizations without going on different sites.

The functionality in portals include: single sign on, customisation, presentation, navigation, interaction, personalization, security, administration and integration (Wege, 2002). Different organizations develop their own portals as means of advertising services that they provide. These organizations range from e-Government to e-Health and electronic schools that deploy their lessons and tutorials over the internet.

The invention of portals created an easy way for businesses to share information and combining their services as one. The design criteria determine the kind of portal each organization is going to have; there is no standard portal architecture. This is due to different types of portals; ranging from user portal, application portal and information portals, etc. All these portals offer different services and their designs should consider the users and what they want to achieve. Architectures are having features that are unique from the other organization architecture. For example, looking at architectures such as Ontroportal, Active life events and Corporate portal. The Ontroportal architecture has a crawler, which is not in other portal architecture such as corporate portal architecture. This is because the other architecture do not have specified need for it. The Corporate portal uses index for categorizing information. There are a number of architectures that have components for categorizing information but use different components compared to the one used in this architecture. There are standards and patterns that are common in the development of portals.

The Model-View-Controller (MVC) pattern is one of the patterns that are being considered in construction of portals according to Gupta and Govil (2010). Figure 2.1 presents an example of a MVC pattern showing the relationship of the components. The Model in the pattern is responsible for responding to the state query and how the back end applications function. The view component allows the controller to select a view. In portals, we can identify this component as the user interface that displays services from back end. The controller defines how the application behaves and selects a view for response. In this work, we adopt the MVC pattern. Users view available services through portal interface which has small windows that are called portlets. Each portlet present a different service (Deacon, 2009; Gupta & Govil, 2010).

Portlets are handled by portlet container that is responsible for user requests. Portlets have shown similarities with servlets. They are both java-based web components, managed by a container, used to generate dynamic content and interact with web clients via a request response paradigm (Goñi, 2008; Nichols & Faulring, 2005; Nichols, Myers & Litwack, 2004). Portlets have some shortcomings and additional features that are not present in servlets e.g. pre-defined modes and states but they only generate fragments. The content generated by a portlet is called fragment.

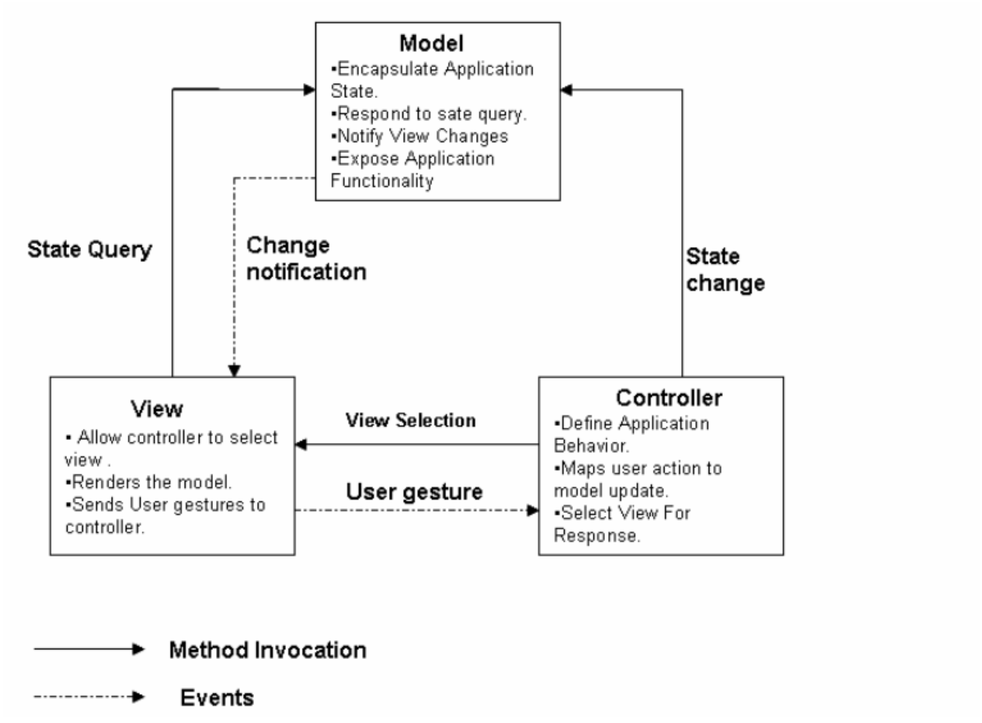
Model
•Encapsulate Application State.
•Respond to sate query.
•Notify View Changes
•Expose Application Functionality
State Query
Change notification
State change
View
• Allow controller to select view .
•Renders the model.
•Sends User gestures to controller.
View Selection
User gesture
Controller
•Define Application Behavior.
•Maps user action to model update.
•Select View For Response.
Method Invocation
Events

Figure 2.1 MVC pattern (Balazs, 2007)

The Web Service for Remote Portlets (WSRP) and Java Specification Request 168 (JSR 168) are amongst the standards and specifications that are used for developing portlets. JSR 168 makes the deployment of Java portlets possible in Java Portal Frameworks without modification.

Most application development architectures consist of three layers and these are database, application logic and interface. So the portal framework offers the fourth layer. This layer is situated between the application logic and the user. The framework specifically presents the software agents in application logic and can be used for co-ordination of loosely coupled services to be a single service by providing the related framework that will integrate them as one. Figure 2.2 gives a sample of a portal architecture with Grid resources.

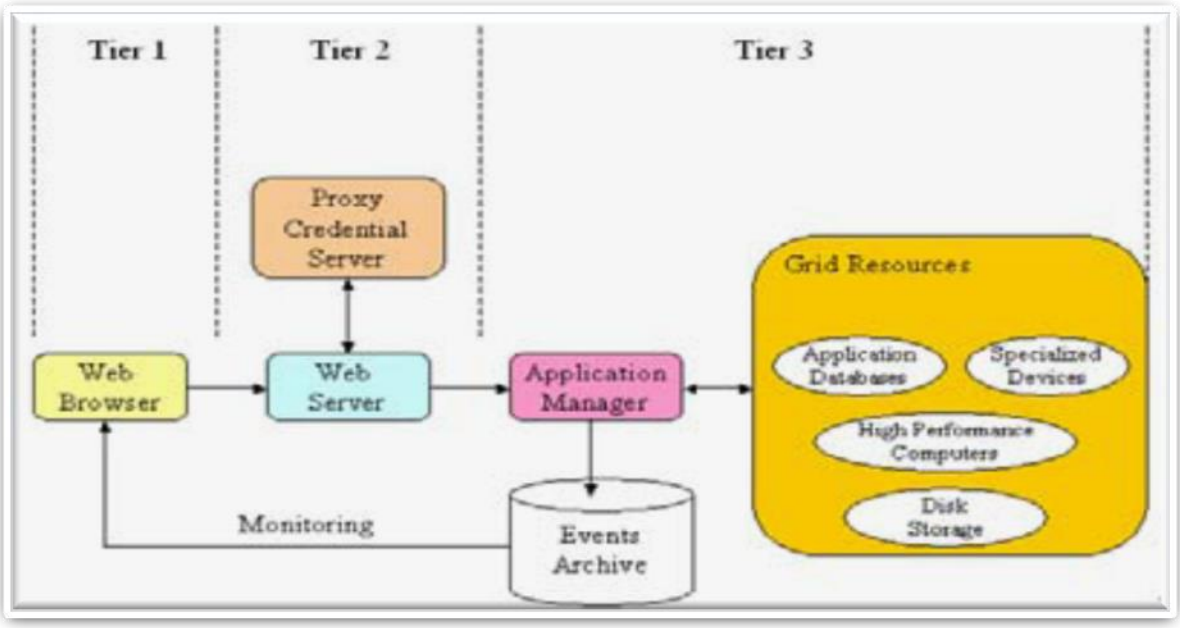

Figure 2.2 A sample of Portal Architecture (Baker, Ong, Allan, & Wang, 2004)

Services in Grid are called Grid services. Grid computing is a distributed computing environment where resources are situated in different physical locations and administrative domains are utilized through virtualization and collective management. Grid computing allows users to access services without knowing where those services are residing. Grid services would be available through a portal. These services would be deployed in order to help organisations to expose their services. This would be advantageous to both the provider and the consumer. The provider would advertise its services and the consumer would have easy access to the services. The GUISET community would use GUISET as the infrastructure in order to render services. A Grid infrastructure is dynamic in nature which means services are deployed at runtime or updated rapidly. The portal interface that would present service for such infrastructure would be required to have a user interface that would adapt to runtime changes.

2.3 User Interface

A user interface (UI) is the front end of any electronic device. It is also known as the human computer interface because users manipulate the system through that interface (Stone, Jarrett, Woodroffe & Minocha, 2005). A user interface consists of an input and an output. The input is where a user can manipulate the system and an output that is where a user can get the results from the manipulation. The user interface is rated on how effective, efficient, learnable and satisfying it is. In each and every system, a user interface should be able to communicate what the system is about and what a user can achieve. The user interface would be usable if it can cater for each and every user's individual need. Friendliness of user interface plays a role in users making use of services in the same portal again. If the users find that it is difficult to use the portal, there is no guarantee that users would be loyal to that portal. There are different types of user interface that ranges from Graphic User Interface (GUI), Web User Interface (WUI), Touch user interface, Command line interface and others (Jørgensen & Myers, 2008). User interface have many limitations which amongst others is their lack of reuse and interoperability (Pietschmann, Voigt & Meissner, 2009). They lack consistence and in most cases the browser of the device determines the type of the interface to be presented based on what the browser can retrieve on the device. Eisenstein, Vanderdonckt and Puerta (2001) argue that applying model base techniques to the development of UI's for mobile computers becomes a solution approach for the lack of consistent and usable user interface. The content presented in the user interface is vital because it reflects what the user has requested or what services an organization is offering.

The user interface has a role of adapting to the changes requested. In most cases the services represented by the user interface are predefined. Adapting to the user request at run time requires the analysis of request and aggregation of components on the fly. The relevance of content in user interface plays a key role in user satisfactory. A personalized interface presents personalized services to the user (Bellas, Fernández & Muiño, 2004; Bellas, Paz, Pan & Díaz, 2008). They can be able to achieve that by looking at the history and profile of the user. Personalized services mitigate the problem that users face of getting services that are not relevant to them. User interface parts are provided as a service and can thus be selected, customized and exchange with respect to the current context (Pietschmann, Voigt & Meissner, 2009). Based on that, we argue that context can be used for the selection of services to appear on a portal at runtime and getting positive results.

There are two ways of generating user interface and these are at design time and at runtime. Generating UI at design time, can tedious to other users because they might not get other services that they are looking for because services are defined during the design. In most portals that have user interfaces that are defined at design time, every user gets the same interface for services. When a user is requesting for a new service it becomes outdated because services are rapidly updated. In portals that allow services to be defined at runtime, a user can specify the service and be able to get it. Getting the services at runtime is not efficient if that service is not what the user was requesting. The relevance of the service at runtime is what most users are looking for.

In an attempt to getting the relevant services to be presented in the UI, the portal should have means of reasoning about the user requests before generating the interfaces. The reasoning over the user request helps in the discovery of appropriate services. There are three factors that influence the relevance of the content represented by a UI:

- the analysis of the user request
- the context of the request
- the selection of services based on the request

During the adaptation of portal interface to the user request, it is important for this process to scale with the number of portlets to be represented and number of request that are made.

Scalability of the interface demonstrates if the portal interface can manage to present the services even with the increase number of users requesting the same service. In portals, scalability can be measured by how the interface performs in bringing the relevant interface. There are a number of factors that affect the scalability in portal interface. The number of portlets in a portal is one of them and the time it takes to respond to a request. Each portlet in the portal interface is presenting a different service and the portal interface act as a presentation layer in GUISET architecture (Candan & Li, 2002).

2.4 GUISET and Service Oriented Architecture

Grid based Utility Infrastructure for Small Medium and Macro Enterprises Enabling Technologies (GUISET) was proposed by Adigun, Emuoyibofarhe and Migiro (2006). This work is one contribution towards its realisation. GUISET is meant to assist the

Small Medium and Macro Enterprise (SMME) by providing them with an e-infrastructure that would enable sharing of resources and collaboration amongst themselves. GUISET plays a role between service clients and service providers by being the mediator. The GUISET infrastructure offers SMME's an opportunity to market and sell their products without having to own the infrastructure or having a direct knowledge of the service provider. A portal interface plays a role of a presentation layer in the GUISET architecture, representing GUISET services. The GUISET services include e-commerce on demand, software as a service and smart search capabilities. GUISET implements Service Oriented Architecture (SOA).

SOA can be defined as a model for building systems based on interaction of services (Akram, Chohan, Wang, Yang & Allan, 2005; Chohan, Akram & Allan, 2005). A service is a software component that enables access to one or more capabilities with prescribed interfaces (Arsanjani, 2004). SOA is mainly concerned with loose coupling among interacting Software Agents and it is now used to support grid-enabled portals. SOA provides cost saving, flexibility and reuse of software components (Akram, Chohan, Wang, Yang & Allan, 2005; Crouchley, Fish, Allan & Chohan, 2004). Figure 2.3 presents a diagram of SOA. Layer three in the diagram is a service layer where services reside. Business process may choose to fund and expose these services and they can be discovered or choreographed into composite services. A service description is a subset of interfaces of a business unit specific component and in some instances an enterprise component. At runtime, the enterprise component is realizable by using functionality provided by their interfaces. Service descriptions are exposed for use. In layer four, services that are exposed in layer three are choreographed in this layer. The services are bundled into a flow and act as a single application. These application support specific business processes, however, there are tools that help to design an application flow. In Figure 2.3 Layer five is Access or Presentation layer. The presentation layer is where software agents are presented as a form of services. There's an increase of convergence of standards such as Web Services for Remote Portlets (WSRP) and Java Specific Request (JSR) 168 that seek to leverage Web services at application interface (Arsanjani, 2004). WSRP is a universal protocol for communication between different portals. JSR 168 is a set of Java APIs that standardize communication between Java portlets and portlet container. GUISET has adopted the SOA mechanism. The SOA characteristics and problems are the same in GUISET.

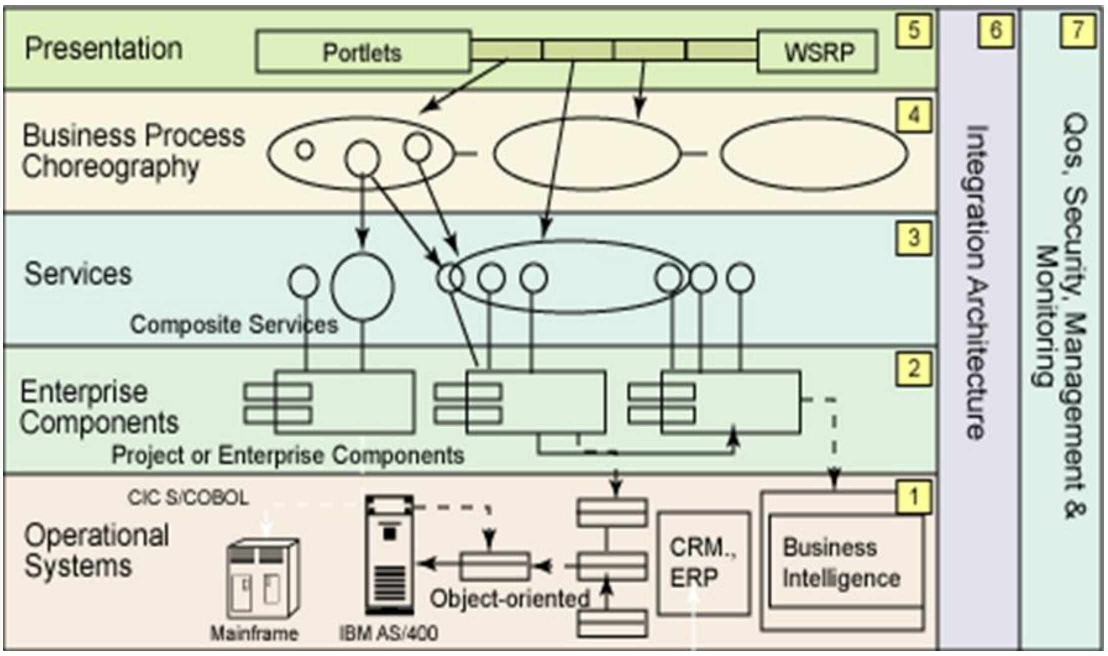

Figure 2.3 Service Oriented Architecture, (Arsanjani, 2004)

Portlets brings presentation in the architecture. In SOA, software components are encapsulated as service. These services are presented as software modules that are accessed by name via an interface in a request reply mode (Natis, 2003). Amongst the principles of SOA, hiding as much of the underlying details of services and the ability to effectively compose services is critical to SOA realization. SOA fundamental blocks are software agents that are self-contained but lack presentation layer (Krafzig, Banke and Slama, 2005). Portals are one of the infrastructures that provide presentation capabilities for the software agent. An adaptive user interface is needed because of service rapid updates in portals (He and Yen, 2007; McKinley, Sadjadi, Kasten, & Cheng, 2004; Savidis and Stephanidis, 2010; Yau and Karim, 2004; Zhao, Ninomiya, Anma, and Okamoto, 2008).

2.5 User Interface adaptation

User Interface adaptation is the seamless change of user interface components based on the user request. In this work, this adaptation is proposed to occur on the fly. This requires the user request to be clear in order to get a relevant content. Clarity in user request, user context and aggregation of components at runtime make user interface adaptation realizable. This results in meeting the user request by presenting the required user interface with relevant content. After the adaptation, it is vital for user interface to be scalable (Tkalcic and Tasic, 2003). It should be scalable with regards to the number of portlets presented in the portal interface and the time taken to present the services (Yu, Benatallah, Casati & Daniel, 2008). Scalable user interface are consistent and are not distorted, they are reusable.

2.6 Summary

This chapter has laid a background of this work. Portal interface acts as a presentation layer to GUISET infrastructure. SOA mechanisms are adopted in GUISET infrastructure. The portal is going to be used by GUISET community to advertise and render services. User interface adaptation towards the user request and presenting relevant content is the basis of this work. In this chapter, the discussion on the importance of adaptive user interface and scalability has also been mentioned.

CHAPTER THREE

LITERATURE REVIEW

3.1 Introduction

The increase in usage of portals in the enterprise environment has led to the portal interface to becoming more usable. Portal interfaces determine the experience a user will get through the portal. The current social impact of the web have shown the web based interfaces are failing to satisfy individual interaction needs of targeting users with different characteristics (Partarakis, Doulgeraki, Leonidis, Antona and Stephanidis, 2009). In Patel and Chaudhary (2009), it is shown that lack of dynamic context in service discovery might result in portal interfaces not presenting relevant services. These are some of the short comings of the portal interface that might influence the usability of portals (Hornbæk and Stage, 2006; Moraga, Calero, Piattini & Diaz, 2007).

Although a lot of research effort has been directed at addressing the shortcomings of portal interface and context awareness in service discovery and presentation of these services at runtime, the results of these attempts still leave a lot to be desired (Repo and Riekki, 2004; Schilit, Adams and Want, 1994; Zhang, Yu and Chin, 2005). This work, as already indicated, is aimed at enhancing the composition of user interface process by introducing adaptation into composition. This chapter covers relevant literature that helps in the descriptive problem analysis phase of our research. Related work on service composition and portal interface, personalisation service discovery, context awareness and open source portals is discussed. From this discussion, the strengths and weaknesses in the current open source portals are identified. This analysis formed the foundation for the solution approach. Our solution approach demonstrates that user context can play a role in improving users getting relevant services in their user interfaces (Hennoste *et al*., 2008; Pietschmann, Voigt and Meissner, 2009; Singh and Wesson, 2006; Stephanidis, 2001). The subsequent sections of this chapter are arranged as follows: Section 3.2 discusses the service composition classes and how these classes are used to address issues of portal user interface. Personalization and service discovery are covered in Section 3.3.The discussion of literature on context awareness and how it influences the service discovery is addressed in Section 3.4. In Section 3.5 covers a discussion on open source portal, evaluating their strengths and weaknesses and Section 3.6 summaries the chapter.

3.2 Service Composition and Component Adaptation

In this section, related work on service composition in web services and runtime component adaptation is discussed. This include: static and dynamic composition and component models.

3.2.1 Static and Dynamic Composition

In the research done by Dustdar and Schreiner (2005) five classes of web services composition were identified, which are:

a) Static and dynamic composition
b) Model driven service composition
c) Declarative service composition
d) Automated and manual composition
e) Context based service discovery and composition

Static service is composed at design time. It uses the original set of conditions when it composes the service. The static interface is customizable and they use user's original set of preferences. Services are limited to those that are described at compiling time. In some instances when updating of services is taking place, the users are interrupted. The content of a static page becomes obsolete in the "on-demand" e-commerce sites because organizations update their services rapidly.

Dynamic composition can overcome some shortcomings of static composition such as lack of timely update and rapid obsolescence of services (Thakkar, Knoblock and Ambite, 2003). Dynamic service composition operates automatically when a user queries for an application at runtime. It involves adaptation of running components at and their functionalities by adding or removing of service components at runtime. Dynamic composition can serve on demand purposes because a user does not have to be offline to update. The runtime definition of services enables an unlimited number of new services to be created from a limited set of service components such that applications are not restricted to the original set of operations specified and envisioned at design time (Alamri, Eid and El Saddik, 2006; Eid, Alamri and El Saddik, 2008). There are several benefits to dynamic service composition as stated in Dustdar and Schreiner (2005); Mennie and Pagurek(2000):

a) Greater flexibility- the customization of software based on the user can be made dynamically without affecting other users on the system.
b) New services can be created at runtime- the application is not restricted on the original set of operations specified at design time.
c) Users are not interrupted during update of applications- the user is not required to be offline. The updating of services is made to be seamless.

d) Unlimited set of services- services are not limited to those that are specified at design time.

In the work done by Alamri Eid and El Saddik (2006) dynamic composition techniques are discussed, depending on the criteria of the system. These are the techniques:

a) Runtime reconfiguration using wrappers
b) Runtime component adaptation
c) Composition language
d) Workflow driven composition techniques
e) Ontology driven web services composition
f) Declarative composition

In this work, we have adopted the runtime component adaptation because it involves adapting components into new components or services by changing the interfaces and implementation behaviour of the component at runtime. Potentially incompatible components become composable in this technique. This enables portal interfaces to be dynamic and adaptive to changes (He and Yen, 2007; McKinley, Sadjadi, Kasten and Cheng, 2004; Savidis and Stephanidis, 2010; Yau and Karim, 2004; Zhao, Ninomiya, Anma and Okamoto, 2008).

“Dynamic user interface or dynamic UIs are portlets or pages that are dynamically created based on the definition of an existing page or portlet definition. A dynamic UI can be launched only by portlet using Dynamic Manager” (IBM, 2011). This shows dynamic interfaces uses a page definition that exists in static pages. In dynamic user interfaces, dynamic content is generated (Alamri, Eid, and El Saddik, 2006; Eid, Alamri, & El Saddik, 2008; Falb *et al*., 2009).

A dynamic content can be achieved through a client side, server side and both. At client side, a page is downloaded and has an embedded java script code which is supported by most browsers. At server side, is when a user is requesting for a web page and the server receives the request and sends an appropriate HTML page which is then displayed. Dynamic content in the context of implementation of portal interfaces means:

a) A portal interface that changes rapidly
b) A portal interface that changes due to responses from a user interface
c) A portal interface using memory that is dynamic

3.2.2 Component Models

The second generation of portals are made up of portlets. Portlets are dynamic in nature and component models. Each portlet presents a different available service and they are independent of each other. Moraga, Calero, Piattini and Diaz (2007) defined the usability of portlets based on user understanding and be able to use them.

Widgets are also component models, they constitute rather self-contained applications on a specific web technology. There is no widely accepted standard specifying a widget component model or description yet, but a movement towards meta-standards can be witnessed. Widgets can constitute either data or functionality in a mashup.

Mashup's main use is the integration of data from different sources and different locations by means of visual composition by the end user. Therefore, they also handle UI integration to some extent. They bundle UI, Action and Service components into a Mashup component that is subjected to composition

(Pietschmann, Voigt & Meissner, 2009). The disadvantage in composition and integration is that these systems are predominantly achieved by custom JavaScript code, which makes development rather time consuming

Object oriented component model that is java based lacks technology independence according to (Pietschmann, Voigt & Meissner, 2009). The system only promotes design-time integration and does not provide means for the context-aware and dynamic UI reconfiguration (Henricksen & Indulska, 2004).

Presentation Integration framework as discussed by Pietschmann, Voigt and Meissner (2009) is presented as a solution in some of the component model's limitations. It is technology independent and it constitutes reuse on the user interface. Its shortcoming is that it allows for the integration on the component level with the help of a composition middleware and platform specific component adapters.

3.3 Personalization and Service Discovery

Personalization is one of the key concepts in portals. A portal acts as gateway of services so when a user accesses a portal, they come across different services. Some of these services might not be of interest to the user. The need for a user to specify what they are interested in makes it easy for the user to access what they are looking for. A user can specify what kind of services to be represented in their own personal account in the portal (Chellappa & Shivendu, 2006). Users are able to customise it and change whatever they do not like.

However, personalised services and interfaces can be obtained by using user profile and user history (Gauch, Speretta, Chandramouli & Micarelli, 2007). The user profile shows the user’s preferred type of service and what the users had specified as their interests. The user history shows the pages or sites that the user usually interact with which might help in the near future.

Most portals have their own personalized pages that a user can create e.g. yahoo has my yahoo where a user gets their own personalized feel (Bellas, Fernández, & Muiño, 2004; Bellas, Paz, Pan & Díaz, 2008). The personalized portals help users to access information in an easy way. There are a number of personal pages that allow users to have personalised services. The limitation in the current portal servers is that they lack generality and adaptation. Bellas, Fernández and Muiño (2004) presented a framework that is structured according to the Model-View- Controller. It is a J2EE based framework that provides generic, adaptable model and controller layers that use the use cases of “my Portal”. In this research work, we are proposing for users to get personalized pages that will demonstrate personalized services.

A service is a well-defined, self-contained function that is not dependent upon the context or state of other services. A service is advertised by service providers through a portal or web site. Portals give organization the opportunity to market themselves and the service they are offering. The service is then discovered, the process of discovering the service requires matching amongst many services. After the service has been discovered it is delivered to the user interface. There are four types of retrieval approaches (Klein & Bernstein, 2004).

Keyword –based retrieval- the search is based on keywords from the service request. This has been discovered to be a poor method to capture the real meaning of the request. Keywords have a problem of recognizing words of the same meaning but treat them differently.

Table-based retrieval- it consists of attributes value pairs that capture service properties. Requests and services are represented with attribute values which are matched. This method is better than the keyword based method.

Concept-based retrieval- it uses ontology for classification and retrieves on types rather than keywords. This method is sufficient but difficult to manage because of the difficulty in defining a consistent ontology of the world.

Deductive retrieval –in this method services are expressed using logic. It consists of deducing which service achieves the functionality described in the query. This method has a problem of high complexity in the matching which result in it operating slowly.

In service discovery, Universal Description Discovery and Integration (UDDI) is used to discover services that are published in it. The goal of UDDI is to describe and discover web services (Oasis, 2004).The central business registry which is used as a naming and directory of services is the core for the UDDI architecture. Services are advertised to the central registry and a user can discover them by query and it uses the result to associate with the service that was requested. There are two functionalities of UDDI, it defines data structure and APIs for publishing service descriptions. It secondly allows the user to query the registry for published descriptions.

The information within the UDDI is classified into four, white pages, yellow pages, green pages and technical model (tModel). White pages is the list of organizations and services they provide and how to contact them. Yellow pages categorizes companies accord to some standard. Green pages consist of information on how a given web service can be invoked. Technical model (tModel) is similar to the yellow pages because it classifies business using standards. The main concept of the tmodel is to make it easy for human being to understand the document because of unstructured data. Broens (2004) improved the service discovery by a mechanism called Context-Aware, Ontology based, and Semantic service discovery (COSS). COSS incorporate context, uses ontologies for common understanding and reasoning and does semantic matchmaking (Goodale, Ludwig, Naylor, Padget & Rana, 2006; Thiagarajan & Stumptner, 2007).

3.4 Context Awareness

There are several definitions for context describing the attributes of context but this work has adopted the following definition from (Abowd *et al*., 1999; Hong, Suh, Kim, & Kim, 2009).

"Context is any information that can be used to characterize the situation of an entity where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including location, time, activities ,and the preferences of each entity."

There are three types of context:

a) The location which entails physical address of an object that consist of a latitude and longitude.
b) Time- has a beginning time and end time
c) Date-specifying current of an object.

User profile has been a basis of content in the olden days, because they reflect user personal taste (Hong, Suh, Kim, & Kim, 2009; Yu et al., 2006; Zhang, Yu, & Chin, 2005) but did not take into consideration factors like location, time and weather etc. These factors play major role when services are discovered. By knowing the location and time a service provider can tell which services are available at that location and at the present time. There are services that are affected by time in certain hours and become unavailable. The limitation of discovering the relevant content has been addressed in several contexts online shops e.g. Amazon applied a collaborative filtering so that the user can have their own personalized store (Linden, Smith, & York, 2003). Xu, Zhu, Zhang and Gu (2008) were also investigating how to personalize the content so they came with a strategy of analyzing the log information and present a personalized content. Contextual information helps the service providers to present more relevant services based on the information they present.

Traditional service mechanism did not give attention to contextual information which could improve the quality of service as functionality in context aware service (CA service) and context aware service discovery. Context awareness can be defined as a property of a system that provides content based on context information. Schilit, Adams and Want (1994) defined three classes of context that can be further categorized:

a) Computing context. It is context information related to context aware system aspects. It entails system context that deals with network traffic, bandwidth and Quality of service, etc.
b) User context. This is context information related to the service requestor. Personal context includes health, mood, activity, etc. Social contexts consists of social relationship and people around, etc.
c) Physical context. Contextual information is related to the physical part of the CA system. Physical context examples location and time, Environmental context e.g. weather, light and Information context e.g. are sport scores, stock quotes.

Some of context entities are also present in non-CA systems (van Kranenburg, Snoeck & Mulder, 2006). In the literature, it has been shown that previous work has focused on context aware delivery of application specific information. Xu, Zhu, Zhang and Gu (2008) argue that if “they log user interaction and perform usage mining using OLAP to discover context depended preference for different information types”. This would result into an adaptive system that would automatically select the most relevant content to the user (He & Yen, 2007; McKinley, Sadjadi, Kasten, & Cheng, 2004; Savidis & Stephanidis, 2010; Yau & Karim, 2004; Zhao, Ninomiya, Anma, & Okamoto, 2008). A context aware application can be adapted to suit the need of a user and the task at hand. Lei, Sow, Davis II, Banavar and Ebling (2002) came up with an application that uses context to predict user’s access to web content and used the prediction to distribute the context to reduce latency and the system is a context aware content distribution system. Context awareness can require less users time because it would not require users to specify some things and it will increase productivity. There is a shortcoming in device heterogeneity which imposes delay in context information to a standard desktop computer. Content distributing systems are solving the latency problem by pre-process and pre-distribute content that suite the user’s device to a nearby network. The approach adopted for the context aware service is by introducing notification dispatching and context aware. Context service should have a clear interface for Quality of Information (QoI) purposes and to express uncertainties (Costa, 2007; Costa, Almeida, Pires & van Sinderen, 2008).

Providing users with service based on preference is important because it helps narrow down services that would be used as option when users request for services (Davis, Tierney & Chang, 2005). Hong, Suh, Kim and Kim (2009) have proposed an agent based framework using user history to provide the personalized services. This framework was proposed with congestion of context history that did not get much attention because it is viewed as collection of past interaction of the user in certain environment and context. Context history can be used to predict the user context e.g. if a user usually goes to movies on Sundays at three o'clock then during that time, movies would be one of the content that would be generated in the user interface(Jong-yi Hong, Suh, & Kim, 2009; Jongyi Hong, Suh, Kim, & Kim, 2009). Deshpande and Karypis (2004) presented an algorithm that would recommend users content, according to the user's history so they use the most used or visited sites that can be generated.

In the literature it has been specified that the user's mood can also have an effect in the content. In this way a user would be granted an interface or services based on how they feel at a particular moment. In the recommendation system offered by De Pessemier, Deryckere and Martens (2009), the content is influenced by the current mood, location and environment.

Today's recommending systems do not take the consideration of the context information they focus on the metadata or the previous consumption behavior to select a content that a user might need (De Pessemier, Deryckere & Martens, 2009; Zhao, Anma, Ninomiya & Okamoto, 2008). A recommendation system has been developed for user generated content. Consumption context is vital in the context information because it determines or gives an idea what the content should be or where the user will utilize the services. A user rating service could also help track which content they prefer more than others. Often than not, portals would generate content that is not relevant in most cases. Some sites would use recommended systems and use the user profile and also user preference information (Gauch, Speretta, Chandramouli & Micarelli, 2007; Kostkova, Diallo & Jawaheer, 2008; Liu, Salem & Rauterberg, 2009). Content Management System (CMS) and portal framework technology usually estimates users interest according to the pages used and this is not always the case. Costagliola, Ferrucci, Fuccella and Zurolo (2008) claimed a need for new metrics that would consider deep level than page visited. They introduced some tools that would help in this new metrics for describing user's behaviour, which produce an XML log and the visibility of portlets in pages when the user were navigating the portal pages.

3.5 Open Source Portals

This section discusses the open source portals, Liferay and Jetspeed-2.

3.5.1 Open Source Portals

Open source portals are portals that are built from open source software. There are a number of definitions for open source software, however, we have adopted the one used in Wheeler (2007) "Open Source Software (OSS) are programs whose licenses give users the freedom to run the program for any purpose, to study and modify the program and to redistribute copies of either the original or modified program without having to pay royalties to previous developers."

There are a number of things that make users to be interested in OSS solutions. They adopt OSS because they are looking for potentially better alternatives to proprietary software and reduced cost. There are OSS challenges that have been investigated in previous researches; the areas of concerned are collaboration issues, organizational and community issues, security, code quality and product quality in general (Viorres *et al*., 2007). In Viorres *et al*. (2007), an investigation was conducted on Human Computer Interaction (HCI). The concern is on the usability of the software, communication, collaboration and more generally interaction issues on OSS communities (Dumas & Redish, 1999).

The open source portals are built from open source portal frameworks. Portal frameworks provides presentation capabilities to the software agents in the SOA and also responsible for the functionality of plugged components by providing the required resources (Akram, Chohan, Wang, Yang & Allan, 2005). The popularity of portals amongst developers has influenced the open portal framework list. These portal frameworks are popular because of some features they bring in development. These are the examples of the portal frameworks given in Akram, Chohan, Wang, Yang and Allan (2005):

a) Sakai 1.5 –broad range in Virtual Research Environment.
b) uPortal – used in Academic Institutes world-wide.
c) GridSphere- the first JSR 168 compliant open source European Portal Frameworks.
d) eXo Platform –it is popular amongst developers.
e) Liferay- it is popular amongst developers, user interface and optional functionality.
f) Stringbeans- it is easy in use.

These portal frameworks address different technologies and framework criteria. However, these portal frameworks differ but there are some core and optional requirement in all of them. The above mentioned portal frameworks were evaluated in Akram, Chohan, Wang, Yang and Allan (2005) based on this criteria:

a) JSR 168 compliant
b) Easy installation

c) Documentation standard

d) Online support

e) Portal Management

f) Portlet resources

g) Performance and scalability

h) Security

i) Technology used

j) Portal features

k) Server dependency

l) WSRP Standard compliant

In Table 3.1 Liferay portal framework has the highest rank followed by eXo platform. In this research we adapted the Liferay portal framework because of its popularity on user interface and its functionality.

Table 3.1: Portal frameworks evaluation(Akram, Chohan, Wang, Yang, & Allan, 2005)

Criteria	Portal Framework					
	Sakai1.5	uPortal	Gridsphere	Exo	Liferay	Stringbeans
JSR-168Compliance	0	5	5	5	5	5
Ease of Installation	3	5	5	5	5	5
Ease of Use	3	5	4	5	4	5
Documentation	2	2	4	3	3	5
Support Services	3	3	4	4	3	5
Administration of Portal	3	5	4	5	4	5
Customisation	4	3	4	3	5	4
Free Useful Portlets	4	3	4	3	5	3
Performance	2	4	3	3	4	3
Security	3	4	3	4	4	4
Technology Use	3	3	4	5	4	3

Portal Features	2	2	3	5	4	2
Server Dependency	3	3	3	4	5	3
WSRP Compliance	0	3	0	3	3	0
Total	35	49	51	57	58	51

3.5.2 Liferay and Jetspeed-2

Liferay Portal is an enterprise portal. It comes with helpful features such as the Content Management System (CMS), (Liferay.com, 2010). An enterprise portal integrates information, application and processes on different organizations. Liferay portal is WSRP producer and consumer, Single Sign On (SSO), and many other latest technologies. Figure 3.1 shows the different technologies that can be integrated to Liferay e.g. Spring and EJB. It is based on the J2EE design and allows the variety of containers. Liferay flexibility in design allows the implementation of business logic with different technologies. Portal pages and portlets in Liferay is easy to customized by the users compared to other frameworks e.g. eXo Platform. The Liferay Portal framework comes with useful portlets which are JSR 168 compliant. Jetspeed -2 is an apache portal. A portal based on Jetspeed can make applications, database information and other data sources available to end users through a single website. Jetspeed provides a security infrastructure so that the information and functions made available to each user can be customised on the user basis or role basis. Jetspeed 2 supports WSRP and is user friendly enhancing user experiment.

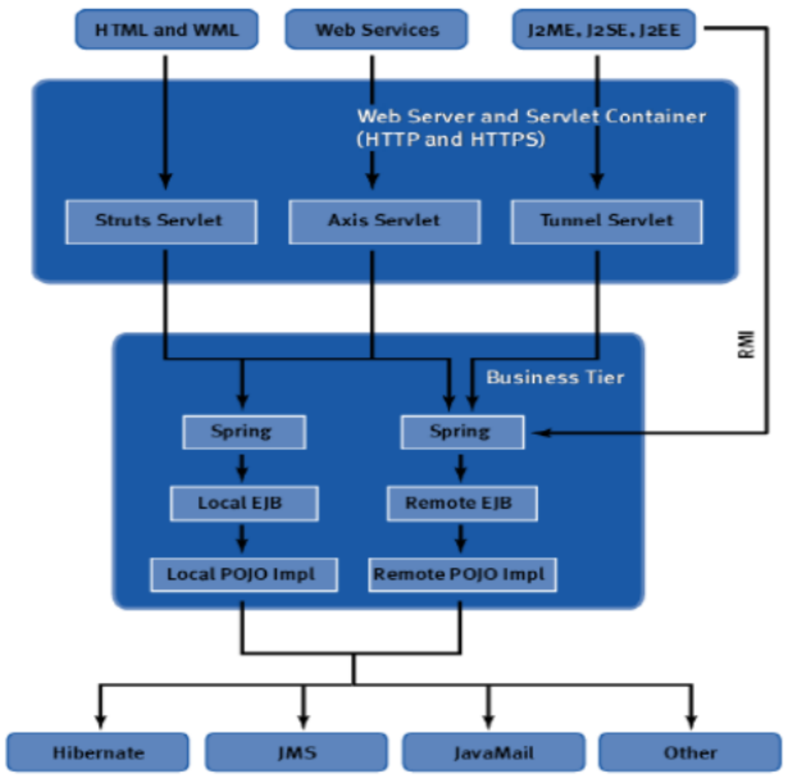

Figure 3.1 Liferay architecture (Akram, Chohan, Wang, Yang, & Allan, 2005; Anupoju, 2012)

In previous research there are some open source framework and containers that are grid-specific portlets but their limitation is they are not WSRP compliant e.g. GridSphere, (Yang *et al*., 2006). Liferay, uPortal and Jetspeed etc. are appropriate for grid enabled portal development; however, they are not grid- enabled portlet. Table 3.2 is demonstrating the evaluation of portal frameworks and containers. The table shows the grid enable portlet portal frameworks that are JSR 168 compliant. Thus this work could not adapt the grid specific portlets portal framework and containers.

Table 3.2: Grid portal development tools (Yang, Dove, Hayes, Calleja, He, Murray-Rust, 2006)

	JSR 168 compliant	WSRP compliant	Grid-specific portlets	Open source
Java CoG 1.2	*	*	–	✓
GPDK	*	*	–	✓
GridSphere 2.1.4	✓	*	✓	✓
GridPort 4.0.1	✓	*	✓	✓
Liferay 3.6.1	✓	✓	*	✓
eXo 2	✓	✓	*	✓
Stringbeans 3.0.1	✓	✓	*	Dual Licences
uPortal 2.5.1	✓	✓	*	✓
OGCE 2	✓	*	✓	✓
Pluto	✓	*	*	✓
Jetspeed -2	✓	✓	*	✓
IBM WebSphere Portal 6.0	✓	✓	*	Free for research

- means only Grid Specific Portlets
* means it does not support feature
✓ means feature is supported

3.6 Summary

In dynamic portal, the need for contextual information is needed in order to get personalized information and also relevant with your current context. Contextual information can increase usability portals because a user will be represented with the preferred services. In a dynamic composition of interface context information would improve the content. Portlet will bring dynamicity in the portal, as they are dynamic in nature. Liferay is the appropriate open portal framework for our research. Liferay is referred to as a content management framework. In this work, the main focus is the content the user would be getting at runtime with context information.

CHAPTER FOUR

THE DESIGN OF A DUI MODEL

4.1 Introduction

This chapter describes the design of the proposed model for dynamic composition of user interfaces. The chapter begins by discussing use case and design requirements of proposed model. The importance of addressing the problem of static user interface is also highlighted. Thereafter, key requirements for dynamic composition of user interface are identified. This is followed by an in-depth discussion of the proposed dynamic composition of user interface model. An explanation of some of the design concepts is also present. The chapter concludes with a summary enlightening the design criteria that has been adapted in this work, which resulted with the design of Dynamic User Interface model (DUI).

4.2 Use Case Scenario and Design Requirements

Portals provide the user-driven mechanism for presenting different services from different vendors using a consolidated user interface. Each organization has its specific goal and specific type of services they offer. Every portal is distinguished by how well the users could take advantage of its offerings and the portal response to user queries. In this work, we have built a dynamic portal interface to meet user's needs in the evolving world of technology.

4.2.1 Use Case Scenario

Consider a user trying to access some services through GUISET portal. The first page the user comes into contact with is the GUISET default page which is also a sign-in page. In the default page the user is presented with services that GUISET portal offers. The user would sign-in to participate as or register to become a member of the GUISET community. After the user has signed in, they open their personalized page. The page displays services that the user had indicated interest in, and also displays services that the user had previously visited. While the portal display different services the user notices that there is a service that they want to render, but is not currently displayed. The user then request for the service that is not displayed, the interface returns services that were available and also the requested service. The portal interface adapts and fits in the portlet displaying the service. The user consumes the service and logs off after finishing using the portal.

4.2.2 Design Requirements

The ability to update or compose interface at runtime and present a relevant content requires a clear description of the interface that must be presented. A clear description is accomplished in the user interface by analysing the user query. The user query can be broken down to check the context of a user. The context of a user in this case is the user profile, user history and preferences (Gauch, Speretta, Chandramouli & Micarelli, 2007). These would give an idea the type of interface the user would likely to take advantage of. By knowing the user context this is helping to discover a narrow scope of interfaces. The context also present the type of layout and theme the user has customised.

The user interface adaptation of components on the fly requires the interface to have attributes that promotes dynamicity. This requires the context of the user to be considered. Aggregators allow the aggregation of different components from different sources that are from different domains to be presented as a single interface.

Aggregators allow comparison and relationship of the sources before presenting a final alternative (Madnick & Siegel, 2001). The services that are presented by the interfaces need to correspond with the clear description from the query analysis, thus requires matchmaking (Goodale, Ludwig, Naylor, Padget & Rana, 2006; Thiagarajan & Stumptner, 2007). Matchmaking the query analysis output and the service discovered will help the aggregator to aggregate the relevant content. In order for the portal to fulfil the goal of this study, it must incorporate capabilities such as:

a) The portal interface should be composing portlets, portlets are dynamic in nature.
b) The portal should allow a personalized page
c) Mechanisms to customize the page
d) The analysis of the user query
e) The presentation of different components of a composite service

4.3 Model Architecture

In this section, we present a Dynamic User Interface (DUI) model that addresses the challenges of interface composition at run time such that contents are rendered based on their relevance (Bykov, 2008; Savidis & Stephanidis, 2010).The DUI uses the service discovery proposed by Goncalves da Silva, Ferreira Pires, and van Sinderen (2009) to address the issue of service discovery challenges mentioned in the literature for dynamic service discovery. The DUI is composed of the following components: user interface, query analyzer, discovery engine with UDDI registry, matchmaker and aggregator, (See Figure 4). These components work together to compose a relevant service even at runtime. These components work to achieve what a user would have specified. Each component is requested to work based on what is requested in the user query.

The proposed model uses query analyser to make sense of a user's request by reducing the user's specification into simple terms. This query analyser is the component that addresses the challenge of making content relevant to the need of the user. In order for relevant content/ services to be discovered there should be a clear understanding of what to look for. The role of the aggregator is to combine components into composite service or content. The aggregation process determines the outcome of services that will be presented at the user interface.

The model considers the issue of presentation of relevant service, usability and reliability which is expected when the user is using a portal (Rubin & Chisnell, 2008; Sauro & Kindlund, 2005; Seffah, Donyaee, Kline & Padda, 2006). The proposed model will address matchmaking in all stages of retrieving a service. It will make sure that the user credentials are considered when a request is made. Matchmaking would reduce the need for users to spend a lot of time searching services because of the lack in verifying if these services go hand in hand with what is requested. This model would also mitigate delay by allowing a user to engage in advance search. The GUISET infrastructure would benefit because the portal will provide a presentation layer. It would help service providers to advertise their services and give their consumers efficient services. The model promotes a dynamic interface by having portlets which are dynamic in nature (IBM, 2011; Pietschmann, Voigt & Meissner, 2009).

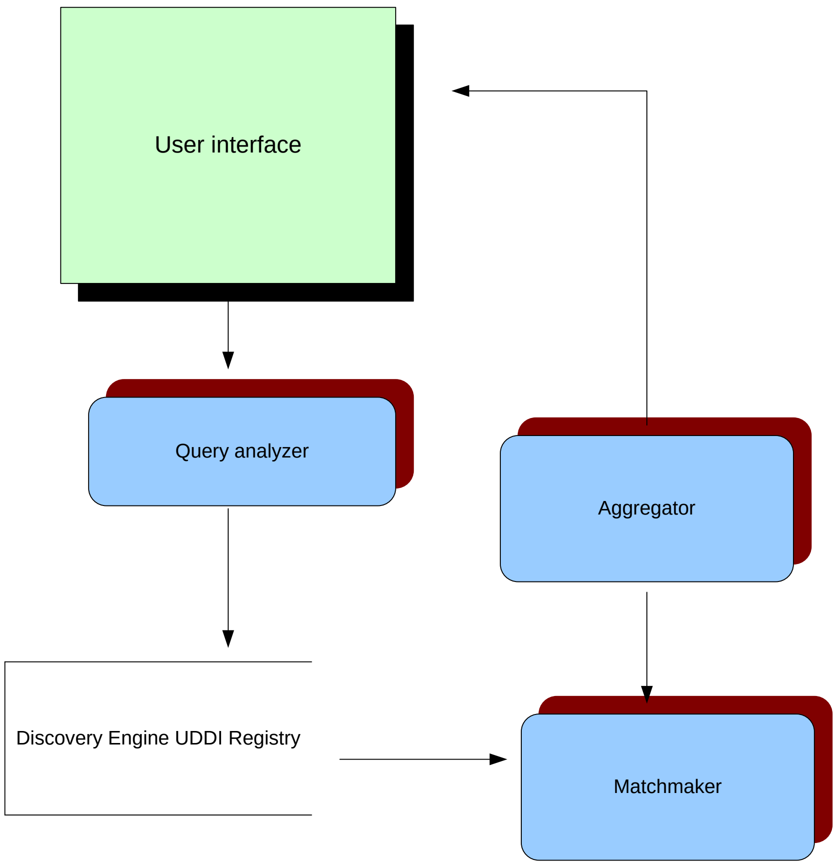

Figure 4.1 Dynamic User Interface model

4.3.1 The Query Analyzer Component

The query analyzer analyses what the user has specified in the query box, the query is usually in a form of a keyword. "Keyword queries are the primary means of retrieving information about a specific entity" (Bazzanella, Stoermer, & Bouquet, 2010). In the literature it has been shown that user behaviour define a context in which the information most likely resides or available at (Teevan, Alvarado, Ackerman, & Karger, 2004). Queries have three categories:

a) Navigational- this query would most likely reach a web site e.g. a query such as Greyhound bus will return a link http:// www.greyhound.com. The navigational query is also known as "know item" and it is used for evaluation of various systems.

b) Informational- this type of query present information that is already in some formatted form, so information query doesn't re-construct any information on the fly. It uses available information in static form.

c) Transactional- the aim for this query is to interact in the web e.g. shopping

In this work the query analyser's aim is to look at the query and filter it and it also considers the three classes of queries. Figure 4.2 presents the query analyzer algorithm showing steps that would be undergone when filtering a user query for better discovery of services. The following pseudo code explains the input and how the output would be generated.

```
Input- Query

Output –Formatted Query

1. Get query
2. Check if the user is a registered member
3. Get profile information
4. Check Format of a query
5. Return formatted query
```

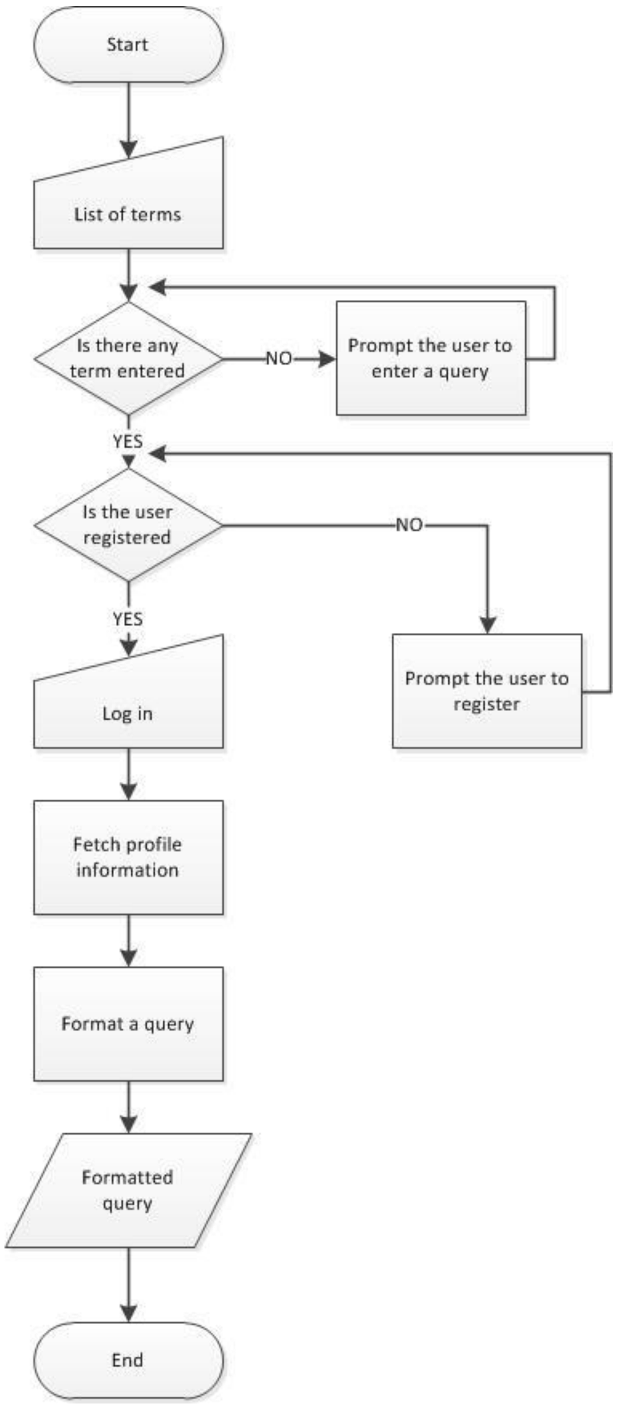
Start
List of terms
Is there any term entered
NO
Prompt the user to enter a query
YES
Is the user registered
NO
Prompt the user to register
YES
Log in
Fetch profile information
Format a query
Formatted query
End

Figure 4.2 Query analyser flowchart

4.3.2 The Aggregator

All the components fulfilling the formatted user query are unified as a single unit in a form of a service. This is achieved by aggregators. These aggregators are able to aggregate different components and present them in a unified manner in the user interface.

Web aggregators are entities that collect information from different source to bring about a single unit or service through the combination of different entities from different sources. Constructing aggregators have become less of a burden because of agent technologies, context sensitive mediator and some extraction tools. The existing tools such as grenouille and cameleon described in Firat, Madnick and Siegel (2000), allow aggregators to collect information from multiple sources without the permission from those sources that the information has been collected from. XML and mediation technology have made it possible for aggregator to extract, compare and analyse information and possibly do automatic comparison. There are three important characteristic of web aggregator: access transparency, contextual transparency and analysis (Madnick & Siegel, 2001). Access to transparency is when an aggregator accesses the data source as a normal user. Contextual transparency performs effective comparison by resolving contextual differences. Analysis is where the information is synthesized according to the post aggregation analysis considering value added information. The organisation whose information can be aggregated is referred to as an aggregate. Since aggregators can also be aggregated, therefore, they can also be aggregates; such aggregators are called Mega aggregator. There are relationship and comparison aggregators. In Table 4.1 examples of aggregators are shown with inter-organisational and intra-organisational perspective. Relationship aggregators aggregate information based on relationship of the entity at hand. Comparison aggregators gather information from different sources to evaluate them. There are aggregators that combine both relationship and comparison features.

Table 4.1: Examples of Aggregator Types

	Comparison	Relationship
Inter-organisational	Comparison of computers prices from alternative suppliers	Consolidation of all one's frequent flyer or financial accounts.
Intra-organisational	Comparison of manufacturing costs from multiple plants.	Consolidation of all information about each customer from the company's separately maintained websites across function and geography

In this work, the aggregator is required to do both the comparison and relationship aggregation. The following flowchart, Figure 4.3 describes how it works. The pseudo code follows the flowchart.

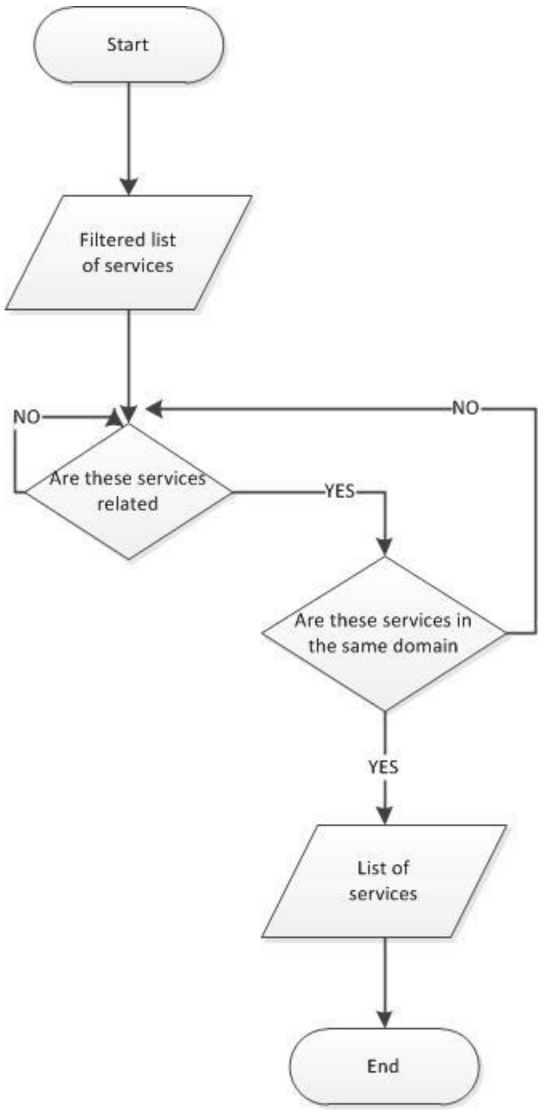
Start
Filtered list of services
NO
NO
Are these services related
YES
Are these services in the same domain
YES
List of services
End

Figure 4.3 Flowchart for Aggregator

Input –Filtered list of services

Output –list of services

1. Get filtered list of services
2. Check if the services have a relationship
3. Check if the services are the same domain
4. Return the list of services

4.3.3 Discovery Engine component

This component takes the output from the query analyser. This output which will be an input in this component will help the discovery engine to look for the exact service that a user is requesting. Discovery is where services are allocated and discovered. Service discovery is the mechanism of finding the appropriate or relevant service. In this component, services are matched using the formatted query. In the discovery of these services, there are contextual factors that affect the kind of services discovered e.g. location, time and weather. These services would be situated in the UDDI. It is a keyword based search. The literature shows there have been different approaches and technologies used. It has been argued that manual discovery of service is error prone and time consuming. Song, Dou and Chen (2011) developed a framework that used a Planning Module and Constraint Satisfaction Problems (CPS) Solving Module. Planning Module takes advantage of service functional attributes and CPS focuses on non-functionality attributes from services that have the same functionality. The output of this component would be a filtered list of services. Below is the pseudo code for discovery engine with formatted query as input. The flowchart (Figure 4.4) follows the pseudo code

```
Input- Formatted Query, Output- Service list

1. Get the formatted query from the query analyzer
2. Extract UDDI query
3. Send query to UDDI
4. Check service functionality
5. Match listed services with formatted query
6. Return filtered service list
```

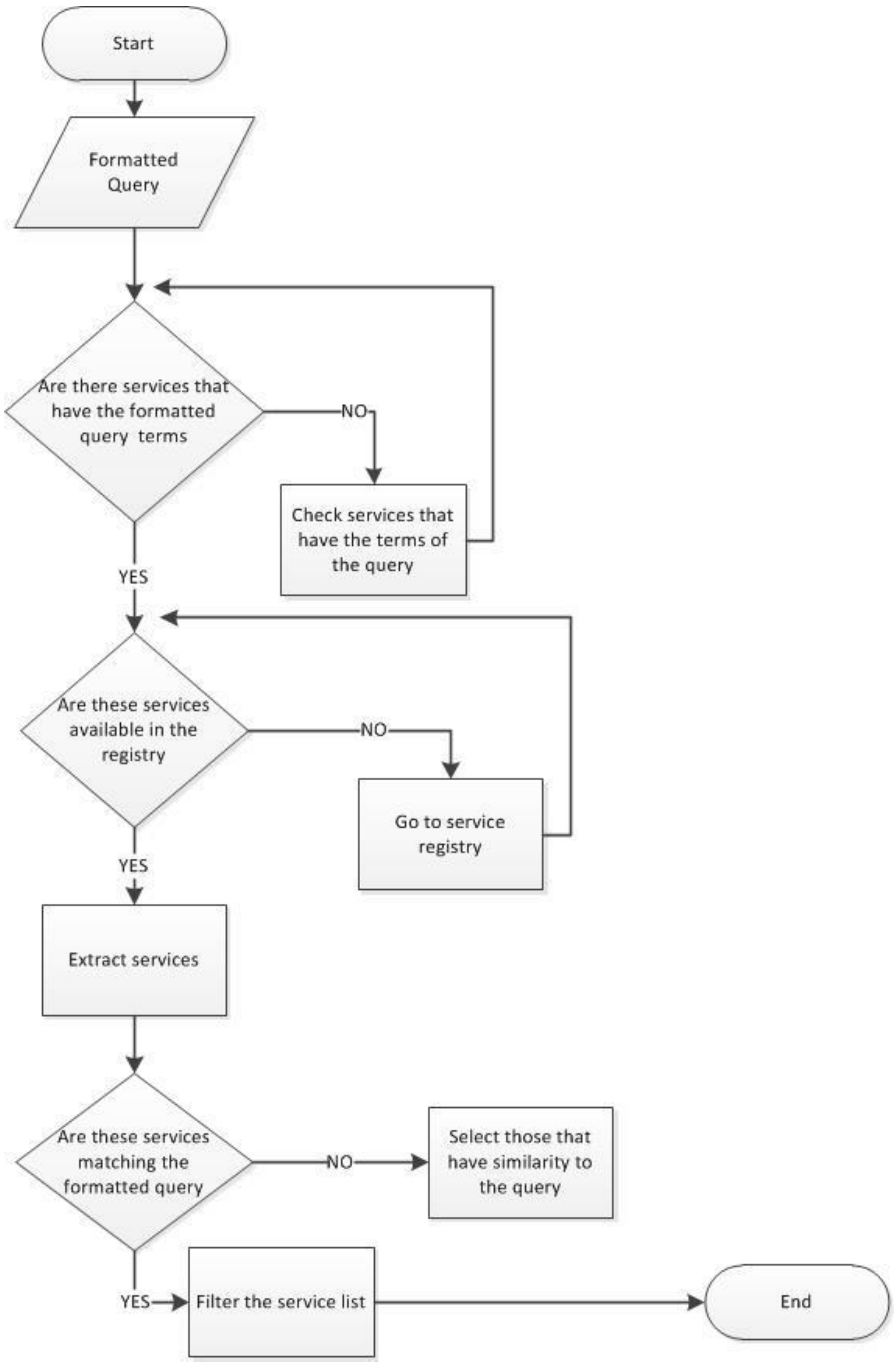
Start
Formatted Query
Are there services that have the formatted query terms
NO
Check services that have the terms of the query
YES
Are these services available in the registry
NO
Go to service registry
YES
Extract services
Are these services matching the formatted query
NO
Select those that have similarity to the query
YES
Filter the service list
End

Figure 4.4 Discovery engine flowchart

4.3.4. Matchmaker component

Matchmaker is used to assist in finding a suitable candidate service. It is composed of reusable components. When you are doing the mechanism of matchmaking, keywords can help in narrowing down the search but they do not offer the description or the compatibility requirements. There are several matchmaking technologies according to Goodale, Ludwig, Naylor, Padget and Rana (2006) that include the following:

a) Ontological reasoning
b) Mathematical reasoning
c) Reputation modeling
d) Textual analysis

Matchmaking has some requirements that should be fulfilled, sufficient input information is needed and the output of the matching should satisfy most of the information that was in the input. In the past, matchmaker was composed of the reasoning engine and the algorithms that helped in definition of the logic of the matching process. Baraka, Caprotti and Schreiner (2005) claimed that most matchmakers are limited to process taxonomies and the pre and post conditions are not considered with their functionalities.

In this work, the matchmaker would match if what has been discovered in the UDDI matches with the request. It would also match if the service matches with the user profile. In a user profile users would have defined their preferences or service that they are interested in. By having this component we are trying to mitigate a problem of getting service that does not match the requirements or the input. In this research, mathematical reasoning and ontological reasoning as others have done was not implemented as it is beyond the scope of this work.

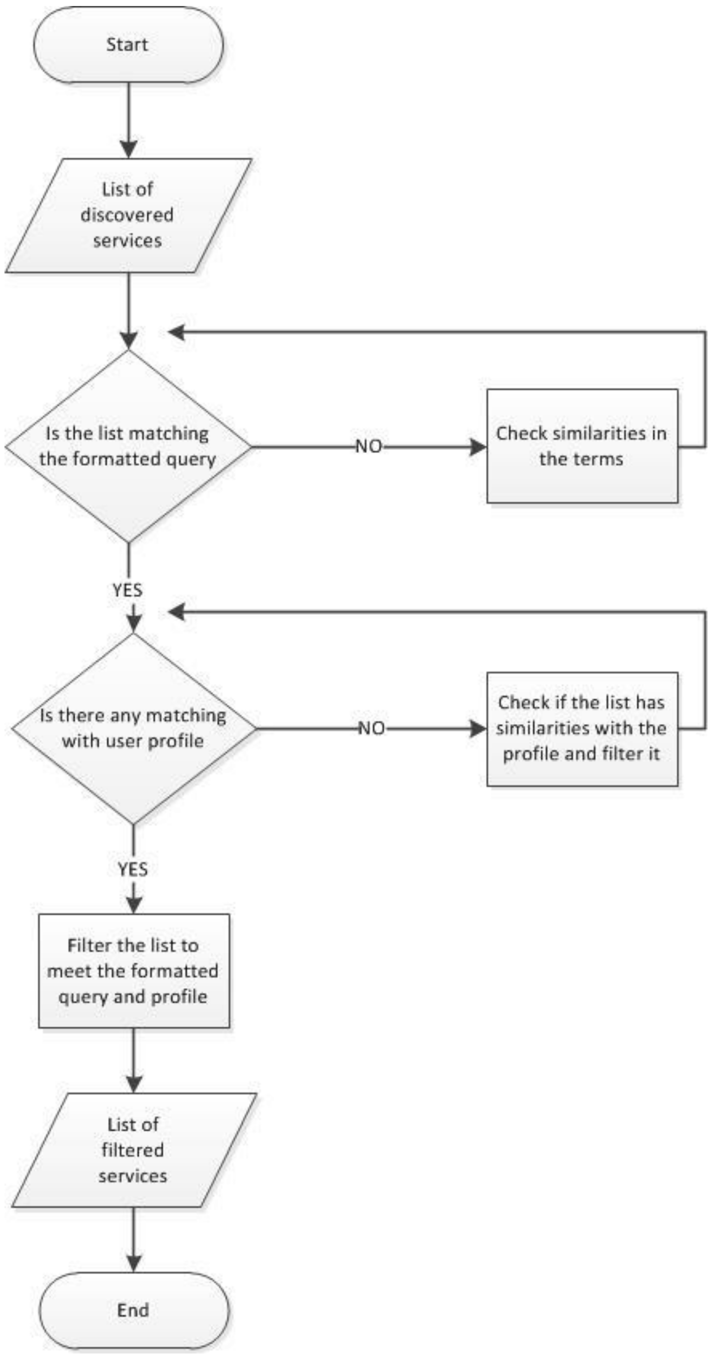
Start
List of discovered services
Is the list matching the formatted query
NO
Check similarities in the terms
YES
Is there any matching with user profile
NO
Check if the list has similarities with the profile and filter it
YES
Filter the list to meet the formatted query and profile
List of filtered services
End

Figure 4.5 Flowchart for matchmaker

Figure 4.5 shows the flowchart for matchmaker. Grid environment has a large amount of dynamic services, so matchmaking is requested to minimize discovery of unnecessary service (Goodale, Ludwig, Naylor, Padget & Rana, 2006; Thiagarajan & Stumptner, 2007).

4.5 Summary

A case scenario of a user attempting to consume GUISET services from a portal resulted in design requirements for a GUISET portal. DUI model was formulated based on the case scenario and the design requirements. Query analyser and aggregator are the core components of the DUI model. The clarification or simplicity of the query has proven to be one of the important components in the model. Most components depend on how the user has specified the query expressing the task they wish to accomplish. Considering the user profile and context history and preferences which in this work is classified as user context also helps in the discovery of relevant services. The aggregator is playing a role of information extraction, comparison and analysis.

CHAPTER FIVE

MODEL PROTOTYPING AND EVALUATION

5.1 Introduction

This chapter describes the prototype of the proposed dynamic user interface model presented in the previous chapter. The main goal of this research was to develop a model for dynamic user interfaces that can adapt to user context based on portal technology. In order to achieve this, design criteria was formulated which resulted in the model presented in Chapter 4. The model, Dynamic User Interface (DUI), has the following components: the query analyzer, discovery engine, matchmaker and the aggregator. The query analyser is a mechanism that is responsible for analyzing user requests. The results of the query analyzer assist in the discovery of services by narrowing down services that would be searched. The discovered services are matched with the output of the query analyzer. These services are aggregated to the interface.

The DUI portal promotes the adaptability of the user interface based on the user request. The DUI model breaks down what the requester has sent into simple terms. To validate the model, experiments were conducted with human subject to benchmark the DUI against Jetspeed-2 portal. Both DUI and Jetspeed-2 portal are enterprise information portals. The experiments were aimed at demonstrating the performance of DUI model and adaptability of user interface at runtime.

This chapter is organized as follows, the assumptions made in the experiments are presented in Section 5.2. Section 5.3 presents, the description of the prototype. The experimentation details for usability with the user perspective are discussed in

Section 5.4. Section 5.5 presents usability evaluation of the proposed model and the questionnaire that was used for data collection. The experimental details for performance in response time perspective are discussed in Section 5.6. The discussion of the results is presented in Section 5.7.

5.2 Basic assumptions of the prototyped model

In developing the prototype, the following assumptions were made in order to delineate what we wanted to evaluate from other factors that may influence the results.

a) Portal interface is not affected by the device being used.
b) The browser is not affecting the interface.
c) The grid infrastructure is running and services are already deployed.

5.3 DUI prototype design

The previous sections have discussed the overall design of the DUI model with its components. This section focuses on the main components of DUI model that is query analyzer and the aggregator. We focus on how these components work with other components. Section 5.3.1 discusses the Use Case Model, sequence diagram and Activity diagram. This section presents the UML models of the DUI Model proposed in this work for Grid services.

5.3.1 UML models

Based on the assumptions presented in Section 5.2, a use case scenario is highlighted through which the DUI prototype can be evaluated. In Figure 5.1, DUI Use case diagram, there is one actor, the user. The user requests for a service in form of a query.

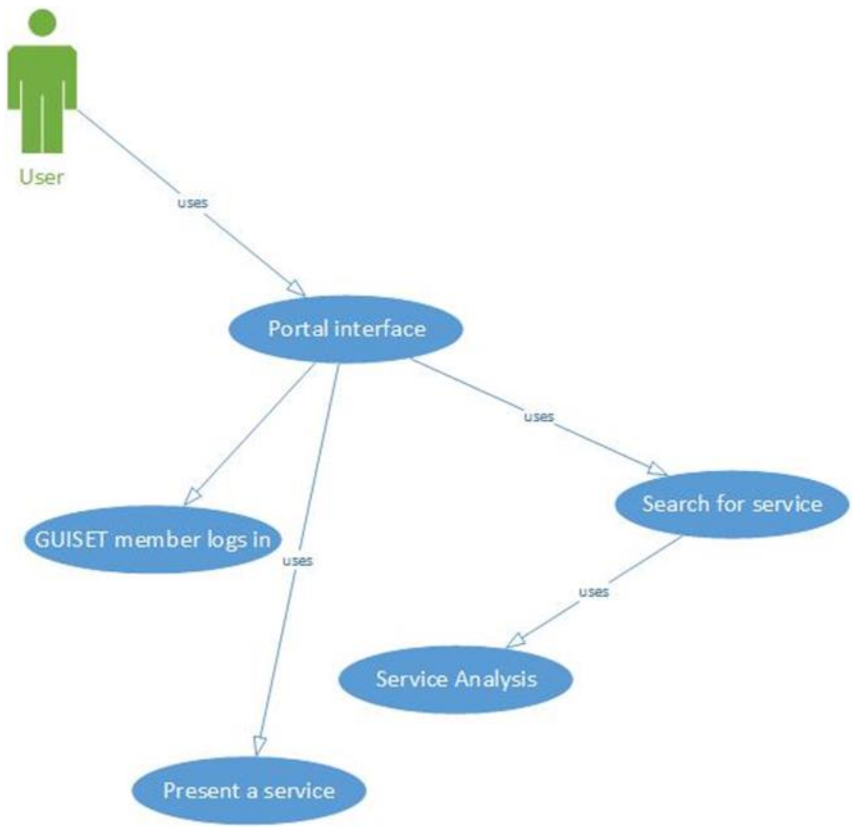

Figure 5.1 DUI Use Case Diagram

When a service is requested, a user interface is called to represent an interface for the service. The aggregator is used to aggregate the different components to compose services. It is responsible for the interface that is represented in the user interface. After the service is requested, the query is analysed to make it easy for the discovery engine to get the services. The list of services from the discovery is matched for it to meet the query analyzer to analyze.

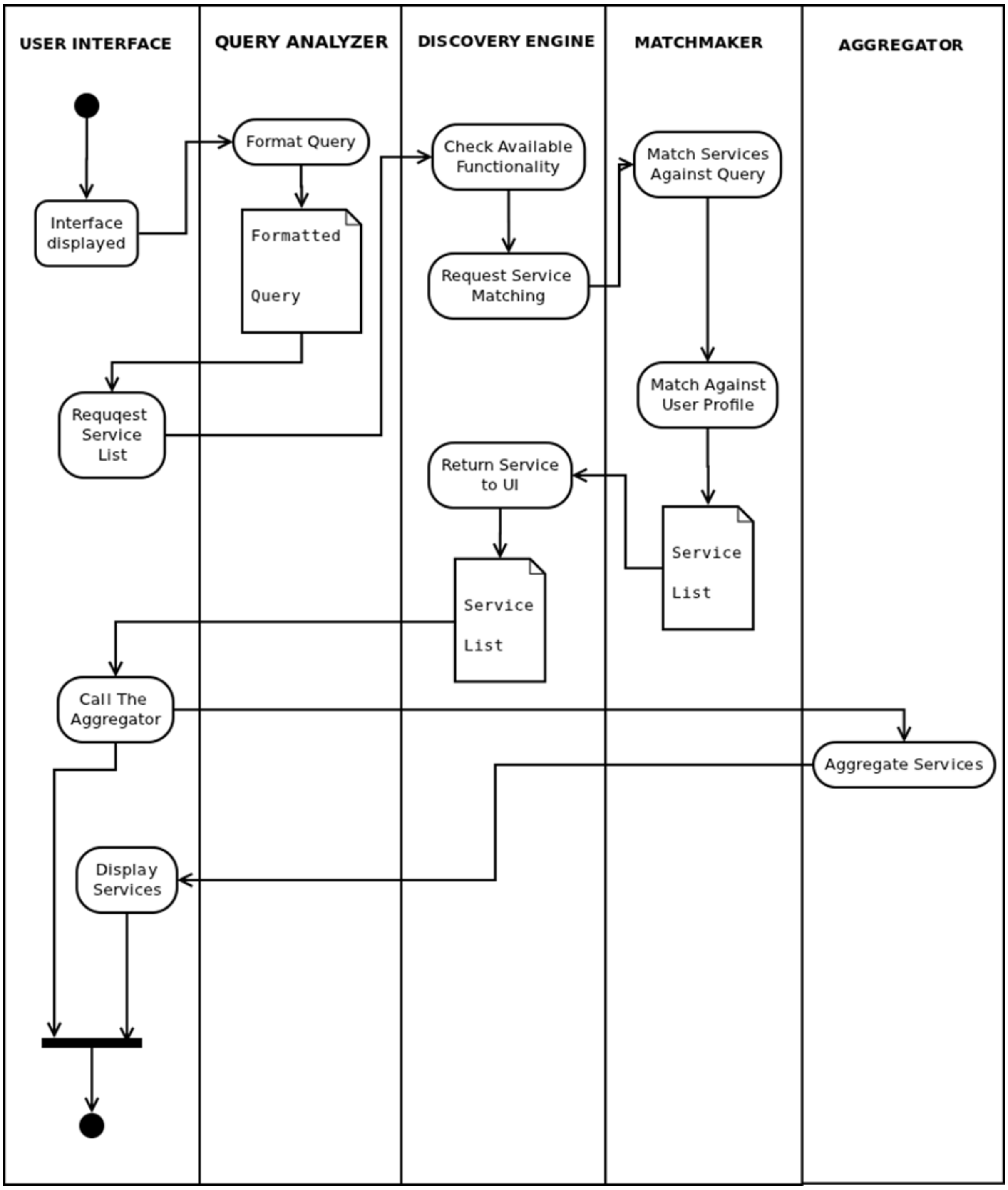

Figure 5.2 DUI Activity diagram

Figure 5.2 shows the state of portal interface before the query and after receiving the queried service. The activity diagram illustrates the different activities that occur during a process of acquiring a service. The above diagram presents what happens in every component of DUI model. The starting point is the user interface display that the user manipulates by stating a query and the end point is also the user interface adapting to the query by displaying the service that was requested. For the adaptation of user interface to the intended user interface, there are class objects that need to be called. The query analysis calls on the query analysis () class in order to breakdown the query. This process occurs in all components of DUI model in order to achieve the service requested.

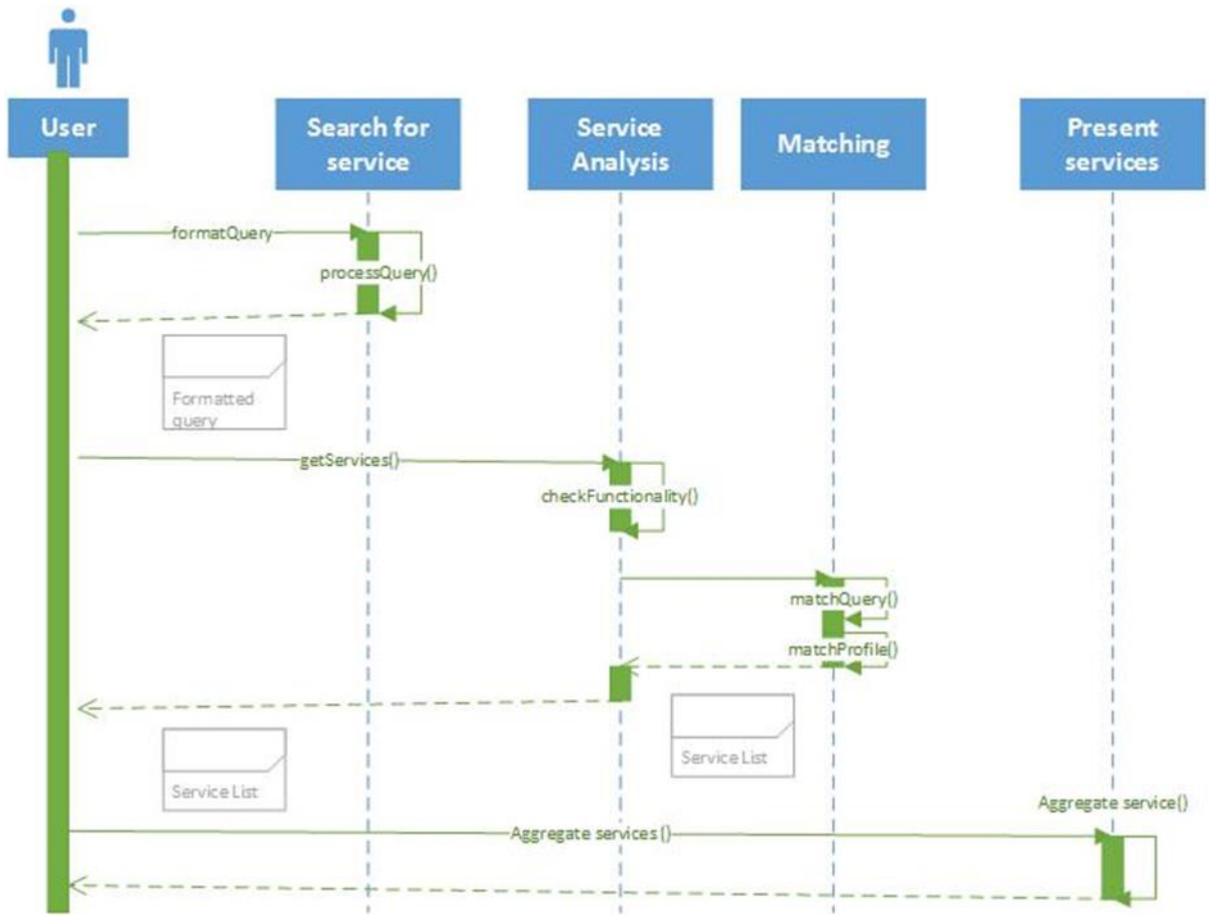

Figure 5.3 Portal interface sequence diagram

In Figure 5.3 DUI sequence diagram shows the flow of messages passing from the user interface (UI) to the aggregator in order to get an interface. These messages are initiated in the user interface by a query; this query is formatted by the query analyzer. This analyzer processes the query and gives back the UI the formatted query. The formatted query is used to call a method of getting the service requested. The discovery engine checks the functionality of the services listed and matches them against the user profile and the query. This is used to call components that make up a service and is represented in the UI.

5.3.2 The DUI prototype Environment

The DUI prototype was implemented in NetBeans 6.9 Integrated Development Environment. The applications were tested on desktop machines running on Windows XP Professional Edition. The ten machines were on Intel Pentium IV processor with a processing speed of 3GHz and 512 MB of RAM. Liferay was used as the main API for developing the DUI prototype. We used Java as a programing language.

Grid services would be available through a portal. These services would be deployed in order to help organizations to expose their services. These organizations would use the GUISET as the infrastructure in order to render services. A Grid infrastructure is dynamic in nature which means the services are deployed at runtime or updated rapidly. The portal interface that would present service for such infrastructure would be required to have a user interface that will adapt to runtime changes.

5.4 Usability experimental Setup

The experimental environment consisted of 10 computers running on Windows XP. There were 60 users from different faculties (four faculties) from the university. Thus resulted in using six groups each consisting of ten students. The first group started to test the Jetspeed-2 portal and tested the DUI portal after the Jetspeed-2, but the next group of students started with DUI portal and finished with Jetspeed-2 to balance out some biasness that may arise from the order which the two interfaces are evaluated by users. The questionnaires were used as an instrument to get feedback from the users (see Appendix). In every set of students the demonstrator explained how the questionnaire is to be filled in. The demonstrator explained how the portal should work. One computer was acting as a server and was not part of the ten computers used by the participants. The computers used an access link in order to access the portals. A user would access the portal and test if it performs as it is expected. After using both portals each user was asked to fill in the questionnaire. The first section of the questionnaire required them to answer some questions that were used to evaluate which class a user belongs to. The table consisted of three columns labelled Never, Sometimes and Very Often. There were two questions that they had to answer based on the experience. The questions were:

- How often do you use computers?
- How often do you use portal interface?

The second part of the questionnaire was composed of four sections for assessing the usability of the DUI Model. These four sections are customization, adaptation after query, user satisfaction with response and portal performance. Each section had a number of questions asking the user to rate two portals under study.

In the customization section, we were interested in checking if the user can further customize the interface and its portlets. By doing so, it might increase the chances of the user having an interface of their choice with their preferred feel and layout. The adaptation after query section examines if the user interface can adapt to the changes they are imposed to react to after a user has parsed a query. This is where the user interface reacts to the current context of the user query. The user satisfaction with response section examines if the user is happy with the response that is presented on the interface after the query has been sent. This section does not only examine the portal interface but also consider the content that is presented. The interface might adapt but the user should be satisfied with the content presented. The last section, the portal performance section, examines the user response time of the portal. Often than not when a system is slow, users panic and often leave the portal without even performing the tasks they wished to perform. The users were asked questions to rate if the portals were fast enough in returning the interfaces they were looking for.

The third section of the questionnaire was similar to the second section in every respect except for the fact that it was for evaluating the Jetspeed-2 interface.

5.4.1 Analysis of Questionnaire

The users were classified into three classes (1) novice, (2) intermittent and (3) experts. The classification justifies how the user will be observing the portals. Table 5.1 shows the results of user classification. These results show that there was no user that has never used computers. 35 out of 60 subjects said they use computers very often and therefore are classified as expert computer users. The remaining 25 sometimes use computers and therefore fall into the intermittent computer user category. In term of usage of portal interface usage we have the following distribution over the subject, 18 of them are novice, 23 are intermittent and 19 are expert users. Even though there were no novice computer users, 18 of the 60 respondents were aware that they are using a portal interface for the first time in our experiments.

Table 5. 1: User classification results

		How often do you use portal interfaces			Total
		Never	Sometimes	Very Often	
How often do you use computers	Sometimes	16	7	2	25
	Very Often	2	16	17	35
Total		18	23	19	60

5.4.2 Usability evaluation results

A statistical test to test the difference between user satisfactions in Jetspeed-2 and DUI was carried out using SPSS version 18. The hypotheses were as follows:

H0: There is no difference between DUI and Jetspeed-2 with respect to customization, adaptation, user satisfaction and user response time.

H1: There is a difference between Jetspeed-2 and DUI with respect to customization, adaptation, user satisfaction and performance.

The results shown in Table 5.2 were obtained. Since the level of significance for all the test is less than 0,05 we rejected the null hypothesis, Ho, with at least 95 % confidence and conclude that there is a difference between DUI and Jetspeed- 2 portal with regards to customization, adaptation, user satisfaction and performance. In Table 5.2, the four sections of the questionnaires were reported as four pairs. Each pair entails statistical results for the Jetspped-2 and DUI portal. Pair 1 indicates the statistical results of customization, Pair 2 shows the statistical results for adaptation, Pair 3 represents the statistical results for user satisfaction and Pair 4 shows portal performance statistical results.

Table 5.2: Statistical results for usability evaluation

		Paired Differences				
		Mean	Std. Deviation	Std. Error Mean	t	Sig. (2-tailed)
Pair 1	Customisation in Jetspeed - Customisation in DUI	-1.42500	2.00650	.25904	-5.501	.000
Pair 2	Adaptation in Jetspeed - Adaptation in DUI	-1.4722222	2.1066585	.2719684	-5.413	.000
Pair 3	User satisfaction in Jetspeed - User satisfaction in DUI	-1.4167	2.3653	.3054	-4.639	.000
Pair 4	Portal performance in Jetspeed - Portal performance in DUI	-1.0250	2.2763	.2939	-3.488	.001

Figure 5.4 demonstrates the results of both DUI and Jetspeed-2 portals. These results are based on the customization of portals. The customization section was examining if a user can be able to understand and be able to customize the portal and portlets. This was based on Diaz, Calero, Piattini and Irastorza (2004) that the best way to use portals is to understand them. The results show a high percentage of users disagreeing with the customization of Jetspeed-2. This shows that the user might be restricted in terms of layout and theme of Jetspeed-2 portal. The DUI portal has shown higher percentage that agrees with its ease portal interface customization. This is because of the flexibility of the portal interface in terms of changing its look and feel based on the user preference. The user is able to add or delete the portlets presenting services that they are not interested in. This can lead to users to be able to use and understand the portal since they will be able to change the layout, theme and the kind of services they wish to see in the portal interface. One of user interface characteristics is its friendliness, in this work friendly user interface is associated with the ease of user interface customization.

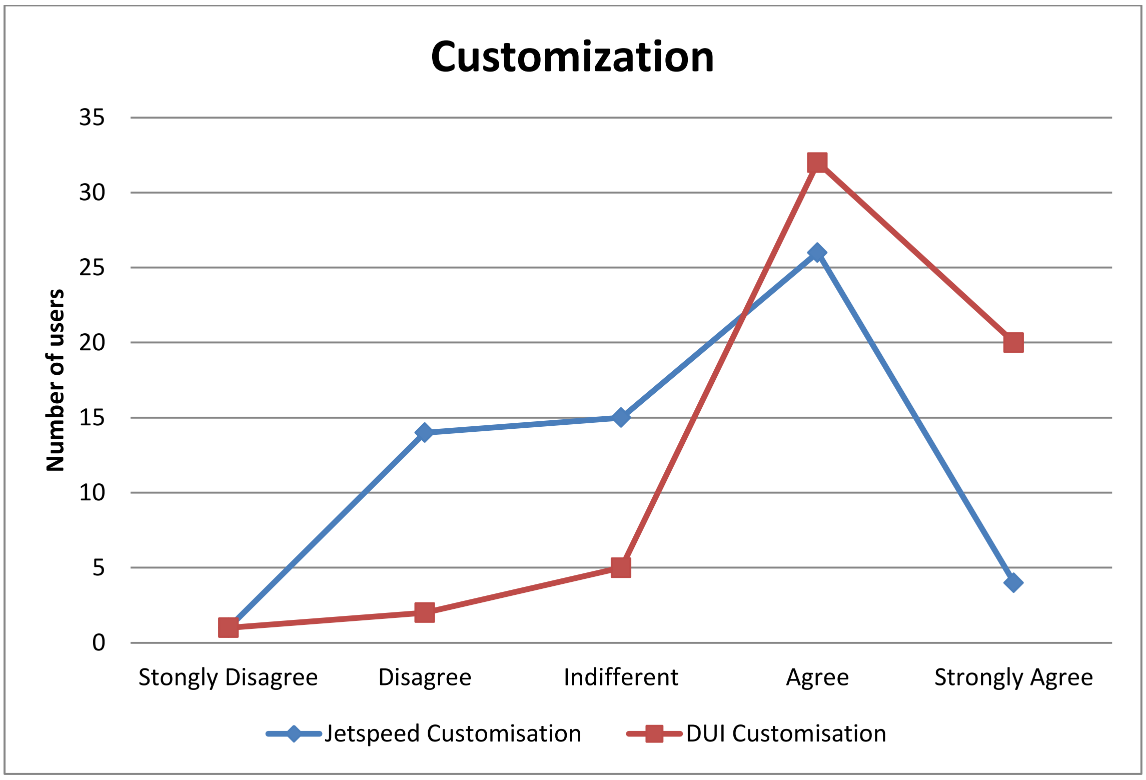

Figure 5.4 Customization of a portals and portlets

Figure 5.5 shows the results of adaptation of an interface after the query. This graph demonstrates if the portal interface changes the user request, calling for an interface at run time. In some cases the interface is defined at design time. The graph shows that the DUI portal interface is able at run time to aggregate different components and presents them as one even though they were not predefined. With the higher percentage of user agreeing that DUI portal adapts better to user queries compared to the Jetspeed-2 portal. This is a result of after the query, the user interface adapted to the runtime changes. The user interface changed to meet user request. Figure 5.5 represent the dynamicity in the DUI Model by the adaptation of the components on the fly.

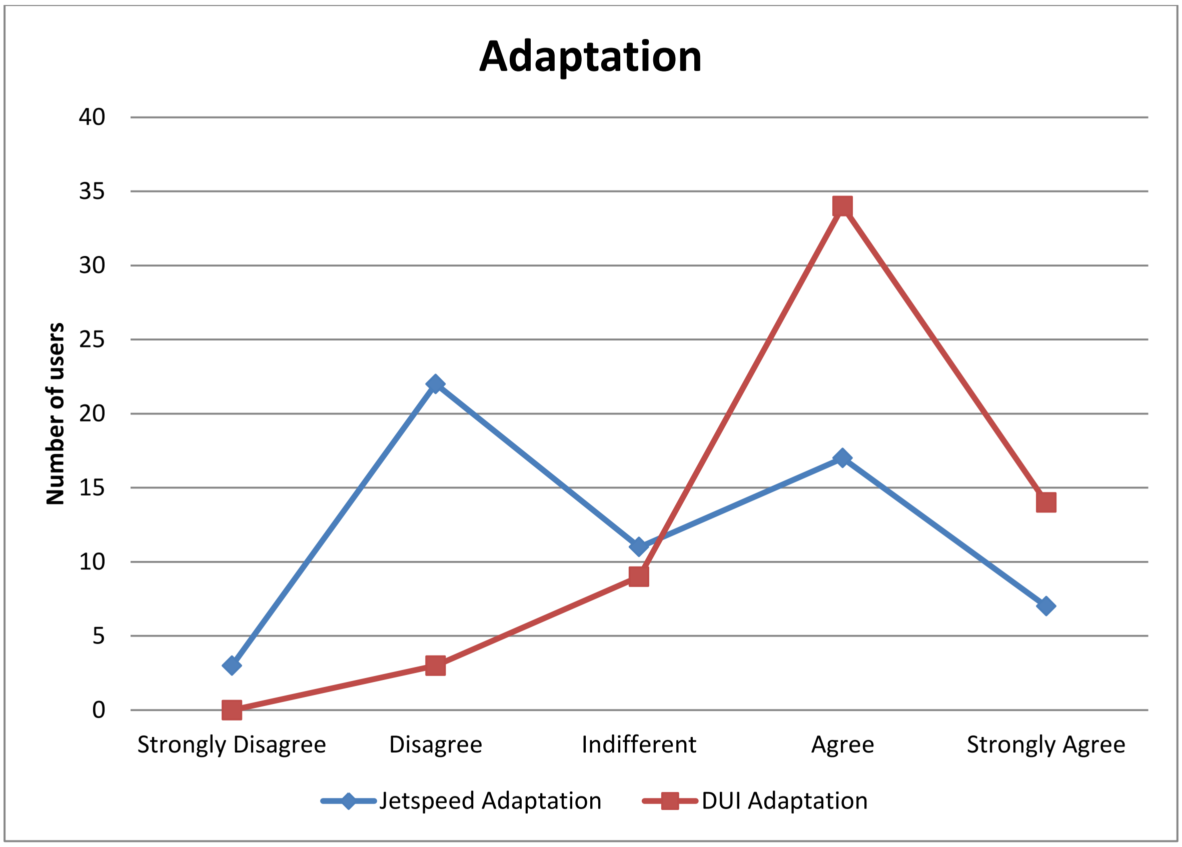

Figure 5.5 Adaptation of portals and portlets

Figure 5.6 presents the users, rating the satisfaction of both the Jetspeed-2 and DUI portal. This category rate of the portal interface meets the expectations of the users when it comes to their capabilities and performance. The user satisfaction is also a part of the usability measurement for the portal. Usability is measured by efficiency, effectiveness and satisfaction according to (Sauro & Kindlund, 2005). The graph demonstrates a high number of users that are satisfied with the DUI portal, compared to the users that are not satisfied with what Jetspeed-2 portal presented. The DUI portal satisfied users requirements based on the adapted user interface after query and the content presented to the user.

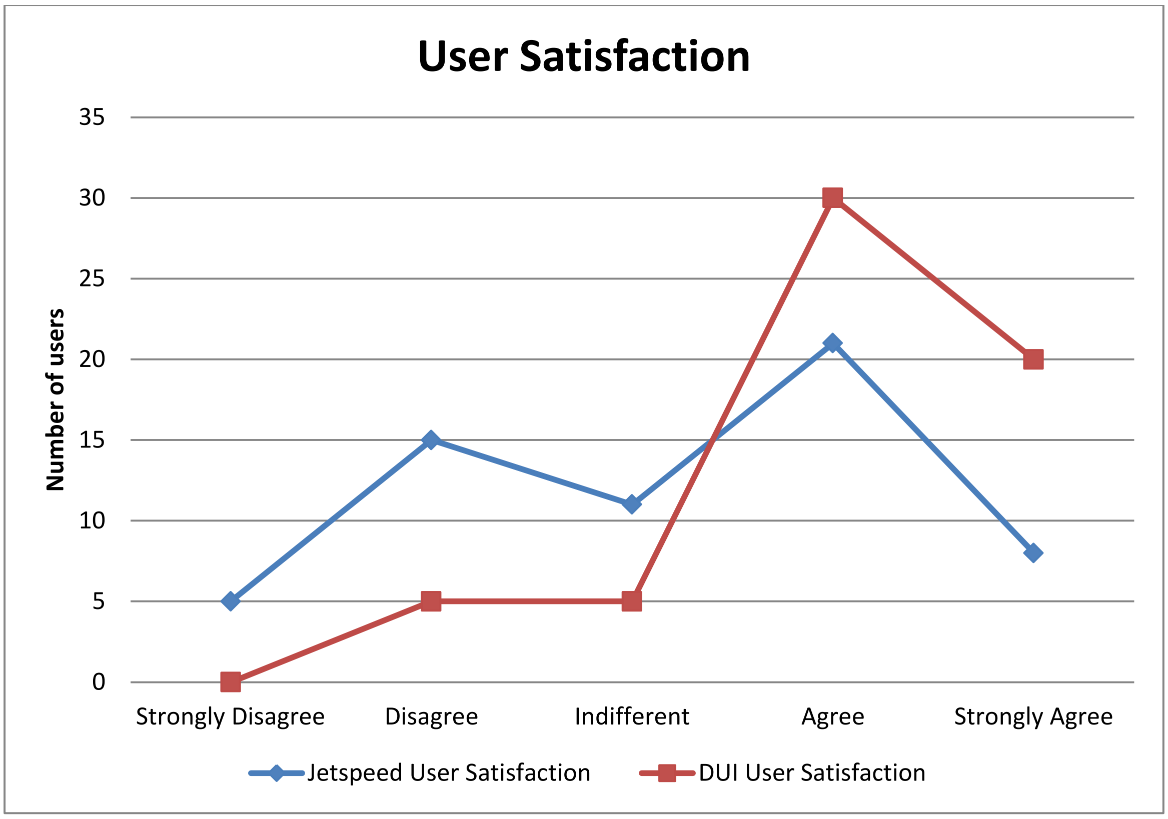

Figure 5.6 User satisfaction with Response

Figure 5.7 demonstrates the effectiveness and efficient of the portal based on human observation. In this graph, it is shown that the DUI portal has the high rate of users strongly agreeing that it is effective and efficient compared to the Jetspeed-2. This is a result of the time the portal takes when performing a task and also the reliability when the users are looking for services. This may also influence the user to visit the portal more often since it won't be time consuming to perform a task. Often than not most users want a system that will be fast enough yet effective. The portal performance graph shows that it is effective and efficient. The effectiveness is measured by the speed on returning a required interface based on human observation. This is influenced by the delay in which the portal takes to load an interface.

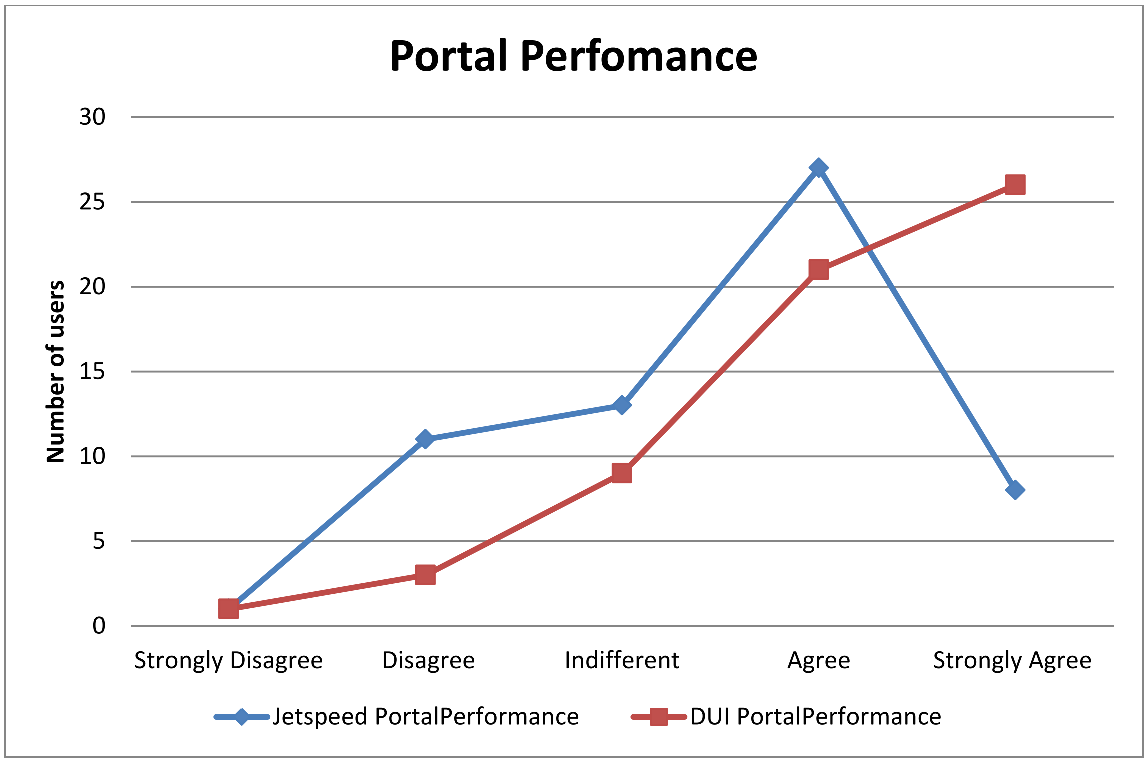

Figure 5.7 Portal Performance

5.5 Technical performance evaluation

The technical performance evaluation of this work consisted of two experiments in which we evaluated the effect of number of portlets with respect to the response time and the number of requests versus response time in the portal. The experimental environment consisted of five computers. The first computer acted as a server and the other computers accessed the portal via a link (See Figure 5.8). These computers were running on Windows XP.

5.5.1 Technical experimental setup

We conducted the experiments by having two portlets for a start and observed how long it took for the portal to respond. This was done ten times and we took the average. We started by using two portlets and we increased each time by one up until there were ten portlets. Each time we increased the number of portlets we measured the response ten times and took the average. We were measuring the performance of the portal by doing so.

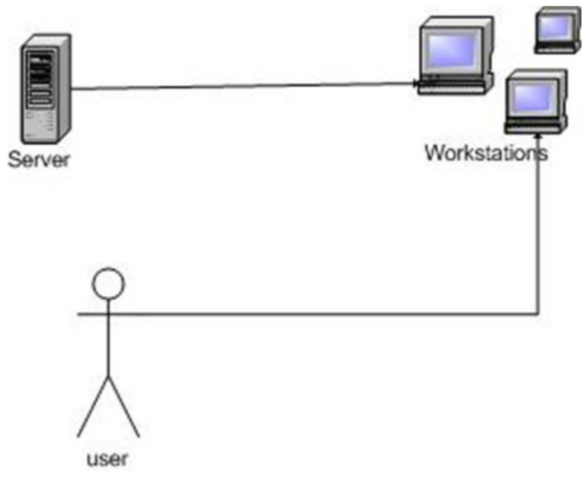

Figure 5.8 Experiment setup

The second set of experiments conducted was based on the number of user requests and the time it took the portal to respond. The users were requesting for the same service. The experiments started with ten user requests; performed ten times and the average was taken. The number of requests started with ten requests up until hundred and twenty request. The number of requests was increased by ten and each time the average of time taken was recorded, after repeating the request ten times. The test was to observe the portal delays. These experiments were conducted in order to understand the technical perspective of the portal performance than human factors only. The experiments also demonstrate how scalable the DUI portal is.

5.5.2 Technical evaluation results

In order to perform a technical evaluation of this work, scalability of the software was measured. The scalability for increasing the number of portlets in the portal is shown by a graph of number of portlets versus the time taken to service the portlets (Table 5.3 and Figure 5.9). Scalability for increasing number of requests in the portal for same services is shown by a graph of the number of request against the time taken to service the requests (Table 5.4 and Figure 5.10)

Figure 5.9 is demonstrating that the number of portlets do not affect the portal performance. The graph shows that there is not much of a significance difference in time taken when number of portlets is increased in a portal. This graph shows the relationship between the number of portlets and time taken in a DUI and Jetspeed-2 portals. The graph illustrates that the DUI is scalable based on the non-effect of the number of portlets present in a portal.

Table 5.3: Measured transaction time for increasing the number of portlets in the portal

NP	TP
2	190.2
3	200.1
4	200.5
5	201.1
6	201.9
7	202.2
8	202.5
9	202.9
10	203.0

NP- number of portlets

TP- time taken to complete in seconds

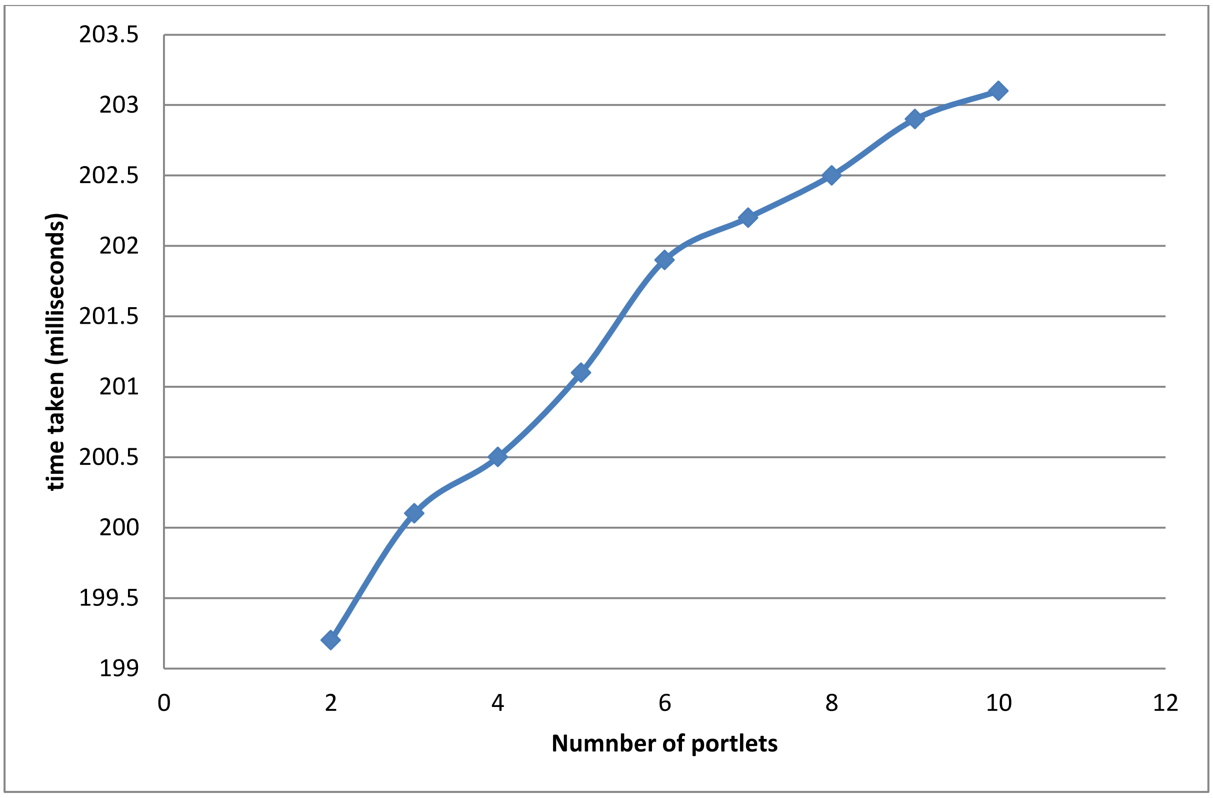

Figure 5.9 Scalability graph for increasing number of portlets in the portal

Figure 5.10 illustrates delay over the number of users. In this graph it demonstrates the performance of the portal when a certain number of requests of a same service are requested. The graph shows that the portal does not have a great deal of delay. In this graph the first set of ten requests had shown isolation compared to the rest. These results resulted to the small number of user request. The results proved the validity of the graph of portal performance showing that the DUI portal can perform much better.

Table 5.4: Measured transaction time for increasing the number of request in the portal

NR	TR
10	322.4
20	322.9
30	323.1
40	323.5
50	323.9
60	323.9
70	324.0
80	324.4
90	325.0
100	325.2

NR- number of requests

TR- time taken for the completion in seconds

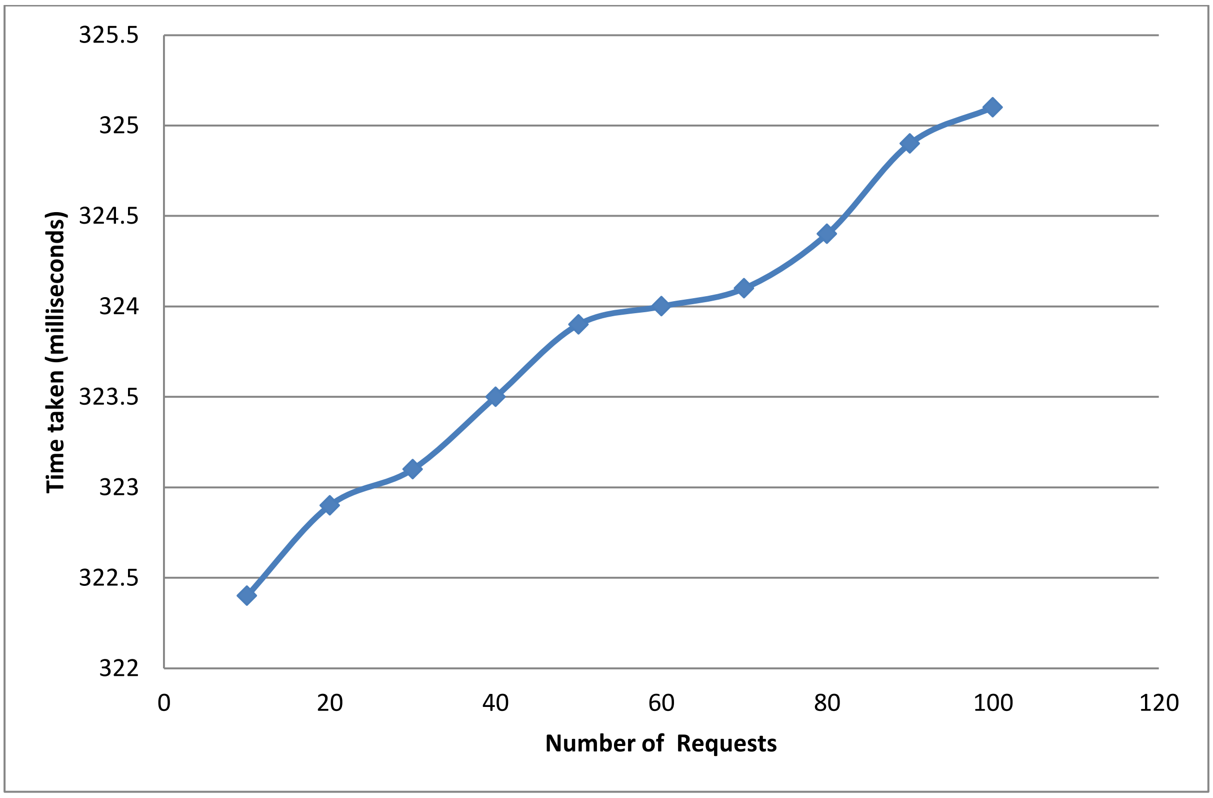

Figure 5.10 Scalability graph for increasing number of service in the portal

5.6 Discussion of results

The results of the experiments conducted on the DUI portal show that the users can understand and are able to use the portal interface with portlets. The customization graph of Figure 5.4 proves the Diaz, Calero, Piattini and Irastorza (2004) statement about usability of portlets and it also demonstrates the customization of portals and portlets which is a vital indication that the user understands portlets and how to use them.

Having an analysis performed by the query analyzer on the fly proved to be user satisfactory based on Figure 5.5 and Figure 5.6. These results show the importance of the query analyzer at runtime. The strategy of analyzing of the log information to give a personalised content used in Zhao, Ninomiya, Anma and Okamoto (2008) was not based on the fly which makes the DUI to be much convenient. The analysis did not consider the history only as suggested by Hong, Suh, Kim and Kim (2009) but also the profile and use preferences. The results show that DUI portal performed better compared to Jetspeed-2 with human factors.

The results shown in Figure 5.9 and Figure 5.10 show the composition of user interface at run time does not affect the scalability of the portal interface. The number of portlets on the portal and the time taken to discover a number of requests at the same time does not affect the scalability of the portal. Both graphs demonstrate scalability.

The satisfaction of user with the adaptation after query is proving the dynamicity in the user interface and also the relevant content which in this work is classified as the user context.

However, the results of the technical evaluation were recorded under a laboratory controlled environment with limited services. In the case of a higher number of services, such as obtainable in real life scenario, the results may differ from those obtain under the laboratory condition. Similarly, the usability evaluations were done in the University community. Thus, chances are that, people who have never used computers (novice) were small in numbers compared to a different environment.

5.7 Summary

This chapter has demonstrated the designing of DUI model and experiments. These experiments were conducted on both the DUI portal and Jetspeed-2 portal. In these experiments, the focus was in two aspects, the human perspective of how the DUI and Jetspeed-2 portal operates based on users experience. The second aspect is the performance on technical perspective. There are three categories of users who have different experience as far as portal knowledge is concerned. Based on the obtained results users felt the DUI portal was both user friendly and usable. Amongst other attributes the personalized user interface in the DUI portal considers user profile, user history and user preference. Customization of a portal and portlets plays a crucial role in understanding portals. The analysis of a query, matchmaking and aggregation of different components to make a service at runtime according to the experimental results yields satisfactory results for the users.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

6.1 Introduction

The purpose of this research work was to propose an architectural model for a dynamic portal interface for GUISET services. GUISET is a platform that the SMMEs are able to interact with by offering services. These services from SMMEs are offered to different types of users. This requires that the portal interface should be able to cater for different category of users from novice to expert users. The portal interface is required to adapt to user request at runtime. The DUI model was constructed to meet those requirements. DynamiCoS framework proposed by Goncalves da Silva, Ferreira Pires and van Sinderen (2009) was used as a basis in designing the DUI model. The DynamiCoS framework aim was to provide end users with automated service discovery and composition. User context and preferences in DynamiCoS framework were not taken into consideration in order to provide a personalised composition. DUI model was prototyped as a proof of concept and to evaluate the DUI model was evaluated. Feasibility of the DUI model was benchmarked with Jetspeed-2 and experiments performed.

This chapter draws a conclusion on the DUI model developed in this research work. In section 6.2 we highlight what we have achieved and also present the summary of this work. Section 6.3 then presents limitations and the future work.

6.2 Conclusion

This work is about proposing an architectural model that could be used in mitigating the obsolete services presented by static portals. The architecture introduces the query analysis that is used at runtime when a service is requested. Query analyzer as a component of this architecture is able to assist by using its output for other components to use and compare. This work has also put emphasis on the role played by user profile, user history and preference in achieving a personalized interface.

The DUI model managed to improve the following:

- User interface customization, the user is able to customize the portal and portlets to the feel and the type of services that they want to engage with. Customization of the user interface promotes personalization within portals.
- User interface adaptation at run time. The interface is able to change based on the user request on the fly.
- User satisfaction. The user is satisfied with the portal functionality based on efficiency and effectiveness.
- Scalability. The portal interface is scalable; it is not affected by the number of portlets in the user interface. The user interface is also not affected by the number of users calling the same service at the same time.

The results have revealed that the DUI model is able to improve a portal interface. The approach that has been adapted for the DUI model enhances the portal performance and presentation of relevant content.

6.3 Limitations and Future Work

The results obtained from the prototype confirmed the effectiveness of the proposed model for the composition of portal interface for Grid services. However, the prototype has a limited number of services that a user was able to search for, this might have limited the experiments. In the future work, having an unlimited number of services would be a primary goal.

This work concentrated on the usability and scalability of the portal interface. Our research pointed relevant issues that needed to be addressed in usability of portals. However, the approach did not consider the Quality of Service (QoS). The QoS mechanism would include checking if the content is reliable and also the recovery of the services that are not available for the interface to be displayed. We would also consider the semantic reasoning in the dynamic service composition by implementing ontologies in future work.

References

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., &Steggles, P. (1999).Towards a better understanding of context and context-awareness.In Handheld and ubiquitous computing, 304–307.

Adigun, M., Emuoyibofarhe, O., & Migiro, S. (2006). Challenges to Access and Opportunity to use SMME enabling Technologies in Africa.In a presentation at 1 st all Africa Technology Diffusion Conference, 14–16.

Akram, A., Chohan, D., Wang, X. D., Yang, X., & Allan, R. (2005). A service oriented architecture for portals using portlets. UK e-Science AHM.

Alamri, A., Eid, M., & El Saddik, A. (2006). Classification of the state-of-the-art dynamic web services composition techniques. International Journal of Web and Grid Services, 2(2), 148–166.

Anupoju, S. B. (2012). Liferay architecture. Liferay Tech Support Liferay Architecture. Retrieved June 21, 2013, from http://liferaysatish.blogspot.com/2012/08/liferay-architecture.html

Arsanjani, A. (2004). Service-oriented modeling and architecture. IBM developer works. Retrieved June 30, 2010 from http://www.ibm.com/developerworks/library/ws-soa-design1/

Baker, M., Ong, H., Allan, R., & Wang, X. D. (2004). Virtual Research in the UK: Advanced Portal Services. UK e-Science AHM.

Balazs, Z. (2007, January 8). A Practical Use of the MVC Pattern.codeproject.com. Retrieved November 10, 2013 from http://www.codeproject.com/Articles/17068/A-Practical-Use-of-the-MVC-Pattern

Baraka, R., Caprotti, O., & Schreiner, W. (2005).A Web registry for publishing and discovering mathematical services. In e-Technology, e-Commerce and e-Service, 2005. EEE’05 Proceedings. The 2005 IEEE International Conference, 190–193.

Bazzanella, B., Stoermer, H., & Bouquet, P. (2010).Searching for individual entities: a query analysis. In Information Reuse and Integration (IRI), 2010 IEEE International Conference, 115–120.

Bellas, F., Fernández, D., & Muiño, A. (2004).A flexible framework for engineering my portals.In Proceedings of the 13th international conference on World Wide Web, 234–243.

Bellas, F., Paz, I., Pan, A., & Díaz, O. (2008). New approaches to portletization of web applications. Handbook of Research on Web Information Systems Quality, 270–285.

Broens, T. (2004). Context-aware, ontology based, semantic service discovery. Master’s thesis, University of Twente, Enschede. The Netherlands, 81–90.

Bykov, V. (2008). Hopscotch: Towards user interface composition. In 1st International Workshop on Academic Software Development Tools and Techniques (WASDeTT-1).

Candan, K. S., & Li, W.-S.(2002). Integration of Database and Internet Technologies for Scalable End-to-end E-commerce Systems. Architectural issues of Web-enabled electronic business, 84.

Chellappa, R. K., &Shivendu, S. (2006). A model of advertiser—portal contracts: Personalization strategies under privacy concerns. *Information Technology and Management*, 7(1), 7–19.

Chohan, D., Akram, A., & Allan, R. (2005).Grid middleware portal infrastructure.In Proceedings of the 3rd international workshop on Middleware for grid computing, 1–6.

Costa, Patr'ıcia Dockhorn (2007, December). Architectural support for context-aware applications: from context models to services platforms. University of Twente, Enschede. Retrieved June 30, 2010 from http://doc.utwente.nl/58357/

Costa, Patricia Dockhorn, Almeida, J. P. A., Pires, L. F., & van Sinderen, M. (2008).Evaluation of a rule-based approach for context-aware services.In Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE, 1–5.

Costagliola, G., Ferrucci, F., Fuccella, V., &Zurolo, L. (2008).Logging and Analyzing User's Interactions in Web Portals. In Web Information Systems and Technologies, 213–229. *Springer*.

Credle, R., Coury, J. and Stimpfle, B. (2006,). WebSphere Portal Best Practises.IBM. Retrieved June 30, 2010 from http://www.redbooks.ibm.com/redpapers/pdfs/redp4100.pdf

Crouchley, R., Fish, A., Allan, R., & Chohan, D. (2004). Virtual Research in the UK: Portal Services for Awareness and Training in e-Science. UK e-Science AHM.

Davis, J., Tierney, A., & Chang, E. (2005).A user adaptable user interface model to support ubiquitous user access to EIS style applications. In Computer Software and Applications Conference, 2005.COMPSAC 2005.29th *Annual International* Vol. 1, 351–358.

De Pessemier,T.,Deryckere,T.,&Martens, L. (2009).Context aware recommendations for user-generated content on a social network site. In Proceedings of the seventh european conference on European interactive television conference 133–136.

Deacon, J. (2009).Model-view-controller (mvc) architecture. Online][Citadoem: 10 de mar\cco de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf. Last access on 10 September 2010

Deshpande, M., &Karypis, G. (2004).Item-based top-n recommendation algorithms.ACM *Transactions on Information Systems* (TOIS), 22(1), 143–177.

Dey, A. K. (2001) Understanding and using context.Personal and ubiquitous computing, 5(1), 4–7.

Diaz, O., Calero, C., Piattini, M., &Irastorza, A. (2004).Portlet usability model.IBM Research Report.RA221 (W0411-084).ICSOC.

Dumas, J. S., &Redish, J. C. (1999).A practical guide to usability testing. Intellect Ltd.

Dustdar, S., & Schreiner, W. (2005). A survey on web services composition*.* *International Journal of Web and Grid Services,* 1(1), 1–30.

Eid, M., Alamri, A., & El Saddik, A. (2008).A reference model for dynamic web service composition systems. *International Journal of Web and Grid Services*, 4(2), 149–168.

Eisenstein, J., Vanderdonckt, J., &Puerta, A. (2001).Applying model-based techniques to the development of UIs for mobile computers. In Proceedings of the 6th international conference on Intelligent user interfaces, 69–76.

Falb, J., Kavaldjian, S., Popp, R., Raneburger, D., Arnautovic, E., &Kaindl, H. (2009). Fully automatic user interface generation from discourse models. In Proceedings of the 14th international conference on Intelligent user interfaces 475–476.

Firat, A., Madnick, S., & Siegel, M. (2000).The cameleon web wrapper engine.In Proceedings of the VLDB2000 Workshop on Technologies for E-Services, 1–9.

Gauch, S., Speretta, M., Chandramouli, A. and Micarelli, A. (2007). User profiles for personalized information access. In The adaptive web, 54–89.Springer.

Goncalves da Silva, E., Ferreira Pires, L., & van Sinderen, M. (2009). Supporting dynamic service composition at runtime based on end-user requirements.

Goñi, M. A. I. (2008). Designing of Portlet-based Web Portals.University of the Basque Country.

Goodale, T., Ludwig, S. A., Naylor, W., Padget, J., &Rana, O. F. (2006). Service-oriented matchmaking and brokerage.In Proceedings of UK e-Science All Hands conference.EPSRC.

Gupta, P., & Govil, M. (2010).MVC design pattern for the multi framework distributed applications using XML, spring and struts framework. *Int J ComputSciEng*, 2(4), 1047–1051.

He, J., & Yen, I.L.(2007). Adaptive user interface generation for web services.In e-Business Engineering, 2007.ICEBE 2007. IEEE International Conference, 536–539.

Hennoste, T., Gerassimenko, O., Kasterpalu, R., Koit, M., Rääbis, A., &Strandson, K. (2008). From Human Communication to Intelligent User Interfaces: Corpora of Spoken Estonian. Proceedings of the Sixth International Language Resources and Evaluation (LREC 2008), European Language Resources Association (ELRA), Marrakech, Morocco.

Henricksen, K., &Indulska, J. (2004).A software engineering framework for context-aware pervasive computing. In Pervasive Computing and Communications, 2004.PerCom 2004. Proceedings of the Second IEEE Annual Conference, 77-86.

Hong, Jong-yi, Suh, E., & Kim, S.-J. (2009). Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4), 8509–8522.

Hong, Jongyi, Suh, E.-H., Kim, J., & Kim, S. (2009). Context-aware system for proactive personalized service based on context history. *Expert Systems with Applications*, 36(4), 7448–7457.

Hornbæk, K., and Stage, J. (2006).The interplay between usability evaluation and user interaction design. *International Journal of Human-Computer Interaction*, 21(2), 117–123.

IBM. (2011). Dynamic user interfaces. WebSphere Portal for z/OS. Retrieved 21June 2013, http://publib.boulder.ibm.com/infocenter/wpdoc/v6r0/index.jsp?topic=/com.ibm.wp.zos.doc/wps/wpsdynui_cpts.html

Jørgensen, A. H., & Myers, B. A. (2008).User interface history. In CHI'08 Extended Abstracts on Human Factors in Computing Systems, 2415–2418.

Klein, M., & Bernstein, A. (2004).Toward high-precision service retrieval*. Internet Computing, IEEE*, 8(1), 30–36.

Kostkova, P., Diallo, G. and Jawaheer, G. (2008).User profiling for semantic browsing in medical digital libraries. In The Semantic Web: Research and Applications, 827–831. Springer.

Krafzig, D., Banke, K., &Slama, D. (2005). Enterprise SOA: service-oriented architecture best practices. Prentice Hall PTR.

Lei, H., Sow, D. M., Davis II, J. S., Banavar, G., &Ebling, M. R. (2002).The design and applications of a context service.ACMSIGMOBILE *Mobile Computing and Communications Review*, 6(4), 45–55.

Liferay.com. (2010).Liferay Portal. Portal, Content, and Collaboration for the Enterprise. - Liferay.com. Retrieved June 21, 2013, from http://www.liferay.com/products/liferay-portal/overview

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing*, IEEE, 7(1), 76–80.

Liu, C.-T., Du, T. C. and Tsai, H.H (2009). A study of the service quality of general portals. *Information & Management,* 46(1), 52–56.

Liu, H., Salem, B. & Rauterberg, M. (2009).A survey on user profile modeling for personalized service delivery systems.In Proceeding of IADIS International Conference on Interfaces and Human Computer Interaction,. 45–51.

Madnick, S. & Siegel, M. (2001).Seizing the opportunity: Exploiting Web aggregation.

Manber, U., Patel, A. & Robison, J. (2000).Experience with personalization of Yahoo! Commun.ACM, 43(8), 35–39. doi:10.1145/345124.345136

McKinley, P. K., Sadjadi, S. M., Kasten, E. P. & Cheng, B. H. (2004).Composing adaptive software. *Computer*, 37(7), 56–64.

Mennie, D., & Pagurek, B. (2000).An architecture to support dynamic composition of service components. In Proceedings of the 5th international workshop on component-oriented programming (wcop 2000).

Moraga, M. Á., Calero, C., Piattini, M., & Diaz, O. (2007). Improving a portlet usability model. *Software Quality Journal*, 15(2), 155–177.

Munćan, M. I. (2009). About portal-based collaborative environments *.Megatrend revija*, 6(2), 291–302.

Natis, Y. V. (2003). Service-oriented architecture scenario. Stamford, CT: Gartner Research. Retrieved November 10, 2013 from www.gartner.com/resources/114300/114358/114358.pdf

Nichols, J. & Faulring, A. (2005). Automatic interface generation and future user interface tools. In ACM CHI 2005 Workshop on the future of user interface design tools.

Nichols, J., Myers, B. A. & Litwack, K. (2004).Improving automatic interface generation with smart templates. In Proceedings of the 9th international conference on Intelligent user interfaces, 286–288.

Oasis, U. (2004). Introduction to UDDI: Important features and functional concepts. Retrieved from www.oasis-open.org

Partarakis, N., Doulgeraki, C., Leonidis, A., Antona, M., & Stephanidis, C. (2009).User interface adaptation of web-based services on the semantic web. In Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments, 711–719. Springer.

Patel, P., & Chaudhary, S. (2009). Context aware semantic service discovery.In Services-II, 2009. SERVICES-2'09. World Conference, 1–8.

Pietschmann, S., Voigt, M. & Meissner, K. (2009). Dynamic composition of service-oriented web user interfaces. In Internet and Web Applications and Services, 2009.ICIW'09. Fourth International Conference, 217–222.

Repo, P. & Riekki, J. (2004). Middleware support for implementing context-aware multimodal user interfaces. In Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia, 221–227.

Rubin, J. & Chisnell, D. (2008). Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests. Wiley.

Sauro, J., and Kindlund, E. (2005).A method to standardize usability metrics into a single score. In Proceedings of the SIGCHI conference on Human factors in computing systems, 401–409.

Savidis, A., and Stephanidis, C. (2010).Software refactoring process for adaptive user-interface composition. In Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems, 19–28.

Schiefer, G., & Kreuder, A. C. (2001). Vertical and horizontal information portals: cooperation models for sector and chain information services. Fritz, M., Kreuder, A.; Schiefer, G. Information Portals and Information Agents for Sector and Chain Information Systems, Report A-01/4, University of Bonn, ILB-Bonn.

Schilit, B., Adams, N., & Want, R. (1994).Context-aware computing applications.In Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop, 85–90.

Seffah, A., Donyaee, M., Kline, R. B. and Padda, H. K. (2006). Usability measurement and metrics: A consolidated model. *Software Quality Journal*, 14(2), 159–178.

Singh, A., & Wesson, J. L. (2006).Using Intelligent User Interfaces to Support Contact Centre Operations.In Proc. 9th Annual SATNAC Conference.

Song, X., Dou, W., & Chen, J. (2011).A workflow framework for intelligent service composition. *Future Generation Computer Systems*, 27(5), 627–636.

Stephanidis, C. (2001). The concept of unified user interfaces. User Interfaces for All-Concepts, Methods, and Tools, 371–388.

Stone, D., Jarrett, C., Woodroffe, M., & Minocha, S. (2005).User interface design and evaluation. Morgan Kaufmann.

Teevan, J., Alvarado, C., Ackerman, M. S., & Karger, D. R. (2004). The perfect search engine is not enough: a study of orienteering behavior in directed search. In Proceedings of the SIGCHI conference on Human factors in computing systems, 415–422.

Thakkar, S., Knoblock, C. A., Ambite, J. L., & others (2003). A view integration approach to dynamic composition of web services.In Proceedings of 2003 ICAPS Workshop on Planning for Web Services.

Thiagarajan, R., & Stumptner, M. (2007). Service composition with consistency-based matchmaking: A CSP-based approach. In Web Services, 2007.ECOWS’07. Fifth European Conference, 23–32.

Tkalčič, M. & Tasič, J. F. (2003).A Framework for a markup language for the definition of Scalable User Interfaces.

Van Kranenburg, H., Snoeck, N., & Mulder, I. (2006).Context aware support targeting plausible and usable results–from reactive, via pro-active to pragmatic systems.

Viorres, N., Xenofon, P., Stavrakis, M., Vlachogiannis, E., Koutsabasis, P., &Darzentas, J. (2007). Major HCI challenges for open source software adoption and development. In Online Communities and Social Computing, 455–464. Springer.

Wege, C. (2002). Portal server technology. *Internet Computing*, IEEE, 6(3), 73–77.

Wheeler, D. A. (2007).How to evaluate open source software/free software (OSS/FS) programs. Retrieved September 16, 2011 http://www.dwheeler. com/oss_fs_eval. html.

Xu, K., Zhu, M., Zhang, D., & Gu, T. (2008). Context-aware content filtering & presentation for pervasive & mobile information systems. In Proceedings of the 1st international conference on ambient media and systems, 20.

Yang, X., Dove, M., Hayes, M., Calleja, M., He, L., & Murray-Rust, P. (2006).Survey of major tools and technologies for grid-enabled portal development.

Yau, S. S., & Karim, F. (2004).An adaptive middleware for context-sensitive communications for real-time applications in ubiquitous computing environments*. Real-Time Systems*, 26(1), 29–61.

Yu, J., Benatallah, B., Casati, F., & Daniel, F. (2008).Understanding mashup development.*Internet Computing, IEEE*, 12(5), 44–52.

Yu, Z., Zhou, X., Zhang, D., Chin, C.-Y., Wang, X., & others. (2006). Supporting context-aware media recommendations for smart phones. Pervasive Computing, IEEE, 5(3), 68–75.

Zhang, D., Yu, Z., & Chin, C. (2005).Context-aware infrastructure for personalized healthcare.*Studies in health technology and informatics,* 117, 154–163.

Zhao, X., Anma, F., Ninomiya, T., & Okamoto, T. (2008).Personalized adaptive content system for context-aware mobile learning. *International Journal of Computer Science and Network Security*, 8(8), 153–161.

Zhao, X., Ninomiya, T., Anma, F., & Okamoto, T. (2008).A context-aware prototype system for adaptive learning content in ubiquitous environment.In IT in Medicine and Education, 2008.ITME 2008. IEEE International Symposium, 164–168.

Appendix: Questionnaire

User categorization questions

	Never	Sometimes	Very Often
1. How often do you use computers			
2. How often do you use portal interface			

Questionnaire for Portal Interface

		0	1	2	3	4	5	6	7	8
	Customization									
1.	I am able to customize the portal									
2.	I am able to customize the portlets									
3.	Information in the customized interface is easy to understand									
4.	The customized interface is easy to use									
	Adaptation after query									
5.	The portal interface is changing to fit the request									
6.	Organization of information after adaptation is easy to understand.									
7.	The adapted interface is easy to use									
	User Satisfaction with Response									
8.	I am satisfied with adapted interface									
9.	I am satisfied with the content on the adapted interface									
	Portal Performance									
10.	Portal speed is fast									
11.	Portal reliability									