E-business framework design using an enhanced Web 2.0 technology

By

Muhammad Sajid Afzal

Faculty of Life Sciences and Computing

Director of studies: Professor Karim Ouazzane

A thesis submitted in partial fulfilment of the requirements of
London Metropolitan University
For the degree of
Doctor of Philosophy

December 2012

Abstract

In the current era of state-of-the-art cutting edge technologies, businesses and organisations are rushing to transform their trade into e-business. The opportunity to utilise e-business improves their chances of gaining a larger market share by maximizing product availability, reducing the day-to-day business activity processing time, and providing related services in a convenient and inexpensive way to their customers.

However in this race, the e-business growth pendulum is only swinging one way, and it is easy to understand the reason for this by observing today's business market. Due to the current financial condition, small business organizations (e.g. local retail shops) cannot afford costly IT systems and the associated maintenance/administration costs, but despite these financial constraints they have an overriding need for computing facilities in order to survive and compete with larger competitors by expanding their businesses.

In this research, Web 2.0 and SOA (Service Oriented Architecture) technology are used to provide a middleware collaboration model between data persistence logic and an operation's requests. This layer helps to overcome the hard-coded service mapping with interface and generic customized workflow problems. This research further provides a mediation platform for request brokers and a high level of abstraction by encapsulating the low level details of the system. These are the most vital requirements to provide a platform, which have the capability of customizing business logic and handle both generic and customized workflows and subsequently to help SMEs (Small- to Medium-sized Enterprises) to convert their businesses to e-business swiftly at minimal cost.

A new W2ASVB (Web 2.0 Architecture for Service and View Brokerage) has been developed, and the validation results confirm that this framework performs efficiently in different conditions, provides on-demand customization of business logic without any data loss as compared to conventional e-business frameworks, and reduces the platform cost drastically due to its shared running environment.

Acknowledgments

I would first of all like to thank my director of studies, Professor Karim Ouazzane, for his unfailing support and encouragement at every step of the way. Without him I would never have come this far. I also feel very privileged to have worked with my supervisors, Professor Hassan Kazemian and Doctor Jun Li. To all of them I owe a great debt of gratitude for their patience, inspiration and friendship. They have taught me a lot about the joy of discovery and investigation that is the heart of research. They have always provided the precise balance of suggestions and criticism.

I am grateful to the Vice Chancellor of London Metropolitan University and the Faculty of Life Sciences and Computing for their generosity in providing me the fee waiver assistance. The Faculty of Life Sciences and Computing has provided an excellent environment for my research; without this rich environment I doubt that many of my ideas would have come to fruition.

Contents

List of figures

List of tables

1. Introduction

1.1 Web 2.0 based e-business framework

The development of the Internet and its integration with Web 2.0 has provided users with sophisticated technologies which ease the process of carrying out their day-to-day activities; this has enabled users to collaborate and interact with systems seamlessly across various domains.

The development and advancement of the Internet has enabled users to complete their tasks in less time and reducing delays associated with conventional methods of interaction. At the same time, it has boosted efficiency and has helped businesses to improve their sales, productivity and economy. However, as mentioned in previous research (Chang, 2006) 'The dynamic, open and convenient web environment not only boosts business potential and the economy but also creates concerns of security, trust, privacy and risks', so users, before utilizing facilities provided for their advantage, should consider and analyze these aspects in order to ensure that they achieve what they require, or get the maximum output from their interactions.

Innovation in Information Systems has changed the way people do business on a global basis, although there are security concerns. During the last two decades, the introduction of IT systems in business has offered new dimensions and challenges to a wide range of different companies. However, existing e-business systems are not affordable for all businesses, especially small retail businesses (such as local shops). This is one of the major problems that need to be addressed. Large businesses tend not to have such concerns, as their main focus is in global expansion of their business.

The gap between large enterprises and small companies in terms of business growth has widened as a result of rapid e-business transformation initiated by

large enterprises, and as a consequence, local retail shops are among the worst affected. The "High-Street Britain: 2015" research report (published by the All-Party Parliamentary Small Shops Group in January 2006), highlighted the essence of small retail shops as:

"The vast majority of contributors agreed that all small shops are important to, and influenced by, economic, social and political trends. The small retail sector is a key driver of: entrepreneurship, employment, skills, local economies, innovation, and sophisticated business networks, as well as accessibility to vital goods and services, diversity, social inclusion and community activities" (Dowd, 2006).

In the context of the relationship between technology and business growth, it can be concluded that the affordability of modern cutting edge technologies has created discrimination in the UK business community, where one giant organization can make significant growth on a daily basis by using sophisticated IT systems; whereas small, independent enterprises (e.g. local shops) are losing their businesses. Financial constraints are the most common cause of non-investment in technology by small businesses; such organizations cannot afford expensive IT systems and the associated maintenance and administration costs due to more limited budgets and investment. In other words, smaller organizations cannot take advantage of new technology because they cannot afford to do so. This technology acquisition gap is now growing faster than ever before.

1.2 Problem analysis

It is always helpful to understand the problem domain before considering the problem in detail. In order to have a better understanding of e-business

frameworks; it is very important to understand the business domain in context, such as the electronic retail industry. This section provides a high-level description of the electronic retail industry, including how this industry works, how individual business organizations run their businesses, what business procedures are common amongst these businesses and which business procedures are different.

Figure 1.1: Electronic retail industry business processes

Any electronic retail business consists of 3 core aspects known as sales, purchase and stock management, as illustrated in Figure 1.1. Retail organizations run their businesses on these three core aspects. For example, any retail company manages the stock of its products, the sale of these products to customers, and the reordering of stock in order to replenish products that have been sold. Regardless of the company size, every retail company has to run this cycle in order for the business to function.

These three core aspects are referred to as core business processes. At the basic level, these core business processes consist of activities and tasks e.g. the "Make

Sale" activity is an activity of sale processing and consists of the following four tasks:

- Entering customer information
- Scanning customer products
- Taking payments
- Saving and printing sale invoice for customer

These tasks are the standard tasks of sale activity, but the execution sequence of these tasks may vary from one company to another. For instance, one retail company might scan customer products first, take payments, record customer details and finally save and print an invoice. While another retail company might perform these tasks in a different execution sequence. The sequence of execution of these tasks is known as workflow.

During analysis, it was noted that most small retail companies follow a common sequence in executing these tasks compared to a few retailers that have a different sequence of execution. Therefore, the common sequence of execution can be considered generic workflows, whereas company-specific execution sequences can be referred to as customized workflows. Figure 1.2 illustrates examples of generic and customized workflows for the "Make Sale" activity.

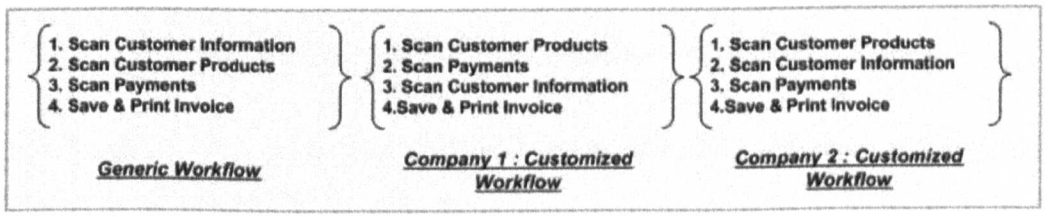


Figure 1.2: Examples of generic and customized workflows

Similarly, most companies share common presentation formats for representing their business data compared to a few that have their own presentation format. In general, this business data presentation format can be referred to as a view. For example, in the case of a customer's sales invoice, the majority of retail companies have their company logo printed at the top of their sales invoices, followed by the shop address, telephone number, information for customers, and then details of products purchased, payment information, VAT information and a "thank you" message at the end. Only a few companies will have some variations in the format of their sales invoices. Therefore, the common view can be referred to as the generic view whereas the company-specific view can be considered a customized view. Figure 1.3 illustrates examples of generic and customized workflows for the "Make Sale" activity.

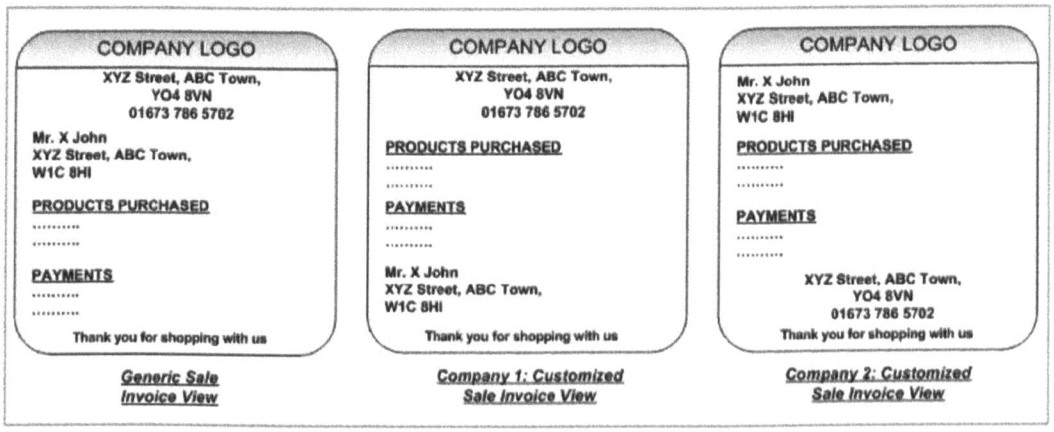


Figure 1.3: Examples of generic and customized sales invoice views

As a result of the discussion presented above on the electronic retail industry, the following important conclusions are drawn:

- All small retail companies follow standard sales, purchase and stock management business procedures.
- The majority of small retail companies share common workflow patterns (i.e. generic workflows) and views (i.e. generic views) during their daily business activities.
- Only few retail companies apply different workflows (i.e. customized workflows) and views (i.e. customized views) during their daily business activities.
- Some retail companies have a combination of both e.g. generic workflows with customized views; customized workflows with generic views; or in special cases, a combination of generic and customized workflows with a combination of generic and customized views.

No business can survive only on its core business processes. It also requires other supporting business processes. Combinations of core and all other supporting processes define the Company Business Process Infrastructure (CBPI). Figure 1.1 represents a typical business process infrastructure for an electronic retail company, where sales, purchase orders and stock management business processes exist in the core process, and other processes such as marketing, accounting, website and logistics support the core business process.

1.2.1 A simple workflow

In a single company system development scenario, information architects usually transform the business activities into services. These services contain the implementation of both data logic and workflows specific to that company. Developers then connect user interfaces (i.e. views) with services to make a

system capable of performing operational requests. An operational request is a request which is initiated by the user via a user interface for data processing purposes.

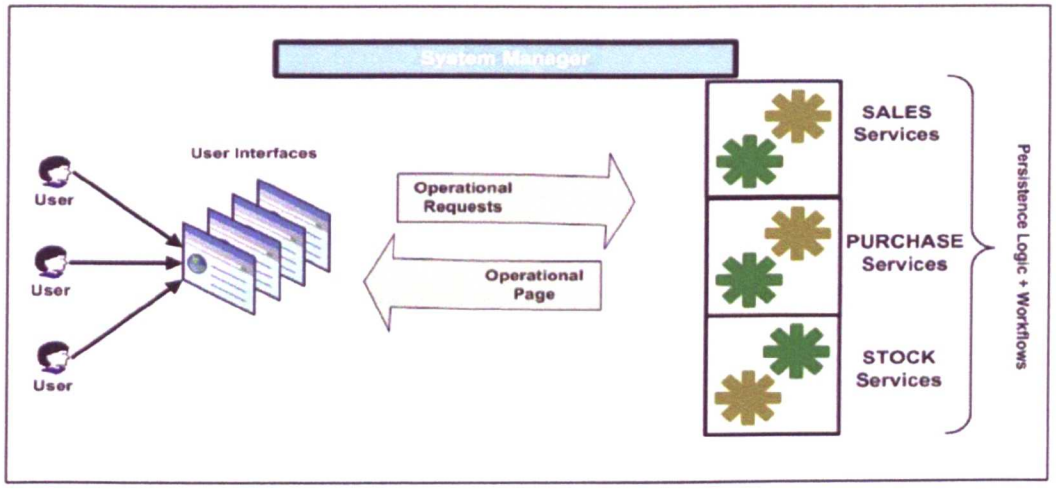


Figure 1.4: A single company base simple work flow

In a traditional Web 2.0 based system, operational requests are usually mapped to services by hard coding the name of the service in a user interface (e.g. an HTML form), as shown in Figure 1.4. After the service finishes dealing with the request, it sends back the response to the user by redirecting it to another user interface e.g. a confirmation page or an error page. When such a system is amended to meet new requirements of a business, new services and user interfaces are added to handle the new operational requests.

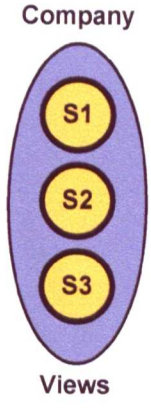


Figure 1.5: Single company business system over service oriented platform

Figure 1.5 illustrates a single company business system infrastructure. It consists of services S1, S2 and S3 and a set of views. Such services encapsulate both logic and the workflow implementation of company-specific business activities. Views (i.e. user interfaces) call these services for processing data and output results are sent back to the user via the user interfaces. In this simple case, explicit request management is not required as user interfaces know which services to call for data processing and services know which user interface is needed to send response back to the user. A web application server performs this request-response routing job automatically.

1.2.2 A complex workflow

A single company system scenario is transformed into a complex state when the idea of developing an enterprise system which not only holds, but also provides, sharing and customization of standard POS business logic is desirable. This enterprise system ideally encapsulates the standard business logic in its core, and facilitates multiple retail companies in executing sales, purchase and stock

management business procedures. Figure 1.6 illustrates the concept of such an enterprise system.

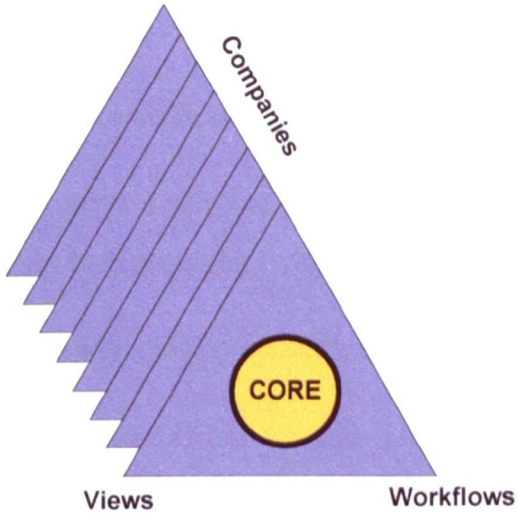


Figure 1.6: Multi-companies e-business systems over one enterprise system

The core of this enterprise system is referred to as the core platform. Initially, to build up this enterprise platform, workflows of each company need to be separated and segregated into one of two categories, namely generic and customized workflows. Subsequently, the core business logic (data persistence logic) that is embedded inside these generic and customized workflows needs to be extracted and will become part of the core platform. In next phase, each company's views need to be separated and segregated into two categories, namely generic views and customized views. This separation of views, workflows and core business logic creates a 3-dimensional system, as shown in Figure 1.6.

To achieve the proposed framework model, the following key questions need to be answered:

- How could the scenario presented be modelled in the proposed framework?
- Are existing Web 2.0 model frameworks capable of solving the above-mentioned scenario?
- How would generic workflows be shared between retail companies?
- How would the customization of these generic workflows be achieved for a retail company (or companies) without affecting others?
- How would a generic view be shared among retail companies?
- How would the customization of these generic views be achieved for a retail company (or companies) without affecting others?
- How would the composite combination of different views and workflows be achieved for a retail company (or companies) without affecting others?

These research questions reflect the complexity of this scenario and need to be addressed in order to find a solution.

1.3 Motivation

This thesis is motivated by the requirement of developing a robust, reliable, efficient and novel framework by using Web 2.0 technology and SOA (Service Oriented Architecture) that will serve as a front and middleware collaboration model between data persistence logic and operational requests. This framework will serve as a mediation platform for request brokers. It will provide a high level of abstraction by encapsulating low level details of the system such as request handling, request mediation, response handling, service loading etc. In order to overcome the hard-coded service mapping to interfaces, with no customizable

business logic and no generic customized workflows problems etc. These are the essential requirements for swiftly converting SME's business into a single e-business platform.

The intended outcome of this research is to provide a platform that will be able to be used as a shared platform for small retail organizations over the Internet. No additional setup, installation, client-side hardware or change in operating system will be required. The system will run over a standard Internet browser and will have the look and feel of a desktop application. On payment of a small annual fee, retailers will have access to an environment to transform their manual day-to-day business processes into e-businesses, and will also be able to manage and customize it according to their individual needs without any major modification in the platform.

1.4 Research Requirments

The following research requirments are posed prior to the main investigation on the basis of the problem analysis.

1.4.1 User request management modelling

Requests can be modelled, located, tracked and then processed based on the request and user profiles to obtain the desired results by using the request management technique. Moreover, to certain extent, this technique handles actions and view management for users' requests. It will allocate a request broker to move the request from a waiting area to the processing area, and start the analysis of a request header. It decomposes the request into sub sections and then allocates the data portion of the request to services modules and finally broking back the responses to the users via the presentation layer.

1.4.2 Workflow modelling

Various generic and customized workflow mechanisms can be identified using different request handling and mediation mechanisms and subsequently the core business logic (i.e. persistence logic) will be embedded within this workflow, in order to overcome the hard coded service mapping between the user interface and business logic.

1.5 Overview of the following chapters

Chapter 2 reviews relevant past work in the field of SOA-based e-business frameworks. It starts with the research of Omar Abbas and Bendiab (2007) on the proposed model framework for defining Web 2.0 components and their relations. It also includes general request handling techniques useful to enhance the performance of SOA bases frameworks.

Chapter 3 describes the conceptual modelling of the framework. It describes the fundamental changes in the proposed framework. It also describes the hypothetical working mechanisms of the proposed model framework within the layer representation.

Chapter 4 describes the details of the proposed model framework architecture. It describes the novel modules involved in complete request workflow and use cases.

Chapter 5 describes the implementation of different modules such as request management, service adapters and system manager. This chapter also discusses request routing techniques used for processing and returning the results back to the desired user.

Chapter 6 describes the validation process of the new e-business framework. This chapter presents the set of instructions used for validation, the pseudo-code of the executor and compares the results of the validation processes.

Chapter 7 offers conclusions, evaluating the overall effectiveness of the proposed model framework described in this study, and a number of directions in which future research could progress.

2. Literature Review

2.1 Introduction

This chapter focuses on an overview of the Web 2.0 SOA standard and the techniques used in enterprise applications. The advantages of SOA-based applications are compared to other similar applications in justification and support of this research, and to discuss the possibility of transferring Web 2.0 SOA.

In an environment where changes are taking place at an increasingly faster rate (Oosterhout, 2006) and organizations face intense rivalry, globalization, and time-to-market pressures (Sambamurthy, 2003), the need for organizational agility and information system agility is considered to be imperative for organizations (Pankaj, 2004). There is considerable interest in service-oriented architecture (SOA) as an agility-building vehicle among IT practitioners. A survey of adoption trends reveals that SOA has become a key consideration for a majority of businesses and is a prominent technology issue in the IT market. It has been proposed as a mechanism to address alignment of IT with business requirements and as a means to achieve IS agility (Bieberstein, 2005), (Kano, 2005). The demand for Web-based solutions and the push toward enterprise-wide integration have led to the use of Web services as a building block for SOA-based applications (Xiong, 2008), (Fan, 2011).

2.2 Brief overview of Web technology

In the last decade or so, Web 2.0 (Murugesan, 2007) ignited innovative and successful web applications such as Blogger, Flicker and YouTube. Although the term "Web 2.0" is regarded as a concept of user-centric web development (Sinton, 2008), most Web 2.0-based software provides its own web services that

are realized in Representational State Transfer (REST) fashion. REST (Mahmood, 2007) is a style of software architecture for distributed hypermedia systems such as the World Wide Web. REST defines a set of architectural principles (Pautasso, 2009) by which one can design web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages (Almeida, 2011).

We are in the process of moving from Web 2.0 to Web 3.0, however in order to appreciate what is new about it and which version is working as a core platform of all web applications, we need to examine the characteristics that defined the previous and current versions.

The second generation of the Web is defined by the empowerment of the end user actively to create content and participate in the Web to display them and relate to other users. The emphasis here is on technologies that enable collaboration such as social networks, RSS Feeds, Blogs, and content publishing services (for images, text and video). Most of these tools are easy to use, which allows virtually anyone to publish a variety of different multimedia content on the Web. RSS feeds in particular allowed for the fragmentation of the "discrete page" concept from Web 1.0. With powerful technologies such as AJAX (Barragans-Martinez, 2010) in Web 2.0, it has become easy to create dynamic and interactive pages that are built by taking information from different sources into a single page, according to the interests of the user. Web 2.0 systems are systems developed using Web 2.0 technologies. These systems offer a rich experience to end-user by providing a desktop-like environment over web browsers on their machines.

Web 2.0 is a second generation design pattern and business model for web applications. The term was first coined by Tim O'Reilly in (O'Reilly, 2005) and states that "Web 2.0 is the business revolution in the computer industry caused by the move to the Internet as platform, and an attempt to understand the rules for success on that new platform. Chief among those rules is this: Build applications that harness network effects to improve as more people use them". The term Web 2.0 is a triggers debate among the research community at present, where some researchers are classifying it as a buzzword, whereas, others are calling it a new concept – a concept which does not change the World Wide Web specification but defines a new way of using it.

Web 2.0 is a new concept that is based on the idea of developing software as service (i.e. SOA) by considering the Internet as a platform and utilizing the power of user contribution towards its improvement (Carey, 2008). In other words, it is a concept of building server-side software as opposed to conventional client-side software where end-user feedback plays a vital role in its evolution. The implementation of software in the form of services on a server-side rather than a client-side is beneficial in terms of providing organization with central management and control without the need of upgrading hardware and software on client machines. Furthermore, using such a technique will not require periodic patch releases to end-users (Clarke, 2010) and (Felber, 2004).

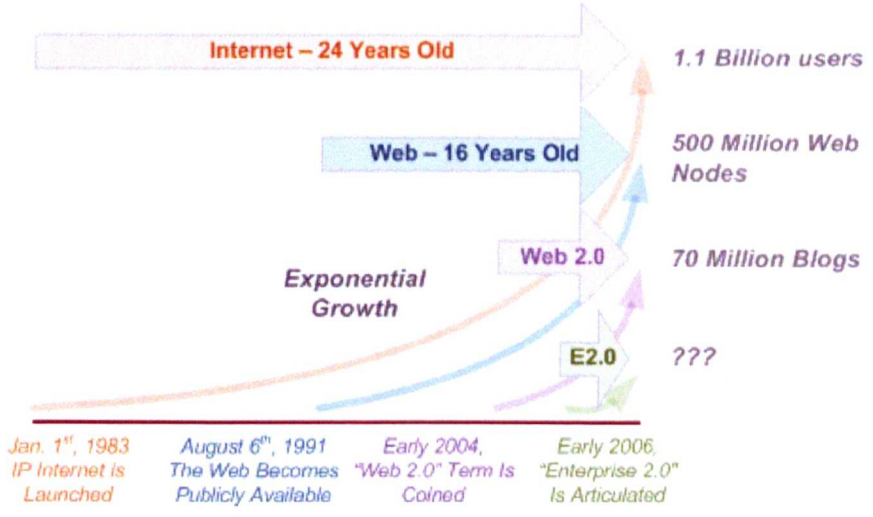


Figure 2.1: Transition of Internet to web 2.0 (Hinchcliffe, 2007)

Figure 2.1 depicts the exponential growth of the Internet in terms of users. According to Internet usage statistics published by Internet World Stats (WorldStats, 2007) nearly 1.7 billion people have been connected with each other on the Internet so far, and these figures are increasing exponentially on a daily basis. Due to the great popularity of the Internet in terms of social networks, nowadays the Internet has moved into a second generation i.e. it is now a self-contained operating platform (Dubney, 2004) – a platform that is technology independent, device independent, always available, accessible to everyone and ever-growing; the more people use it. Software enterprise has realized the power of this new emerging platform and the production of Web 2.0 systems is a current "hot topic". Interactive encyclopaedia, Blogs, and Mash-ups are example applications of the Web2.0 era (Dubney, 2004).

"Web 2.0 is a transformation of web-applications from information to services of contents and functionality, thus becoming computing platforms serving end-users. It promotes concept of community-based collaborative environment and

hosted services – such as social networking sites, wikis, and folksonomies – which aim to facilitate collaboration and sharing between users" (Web 2.0, 2007).

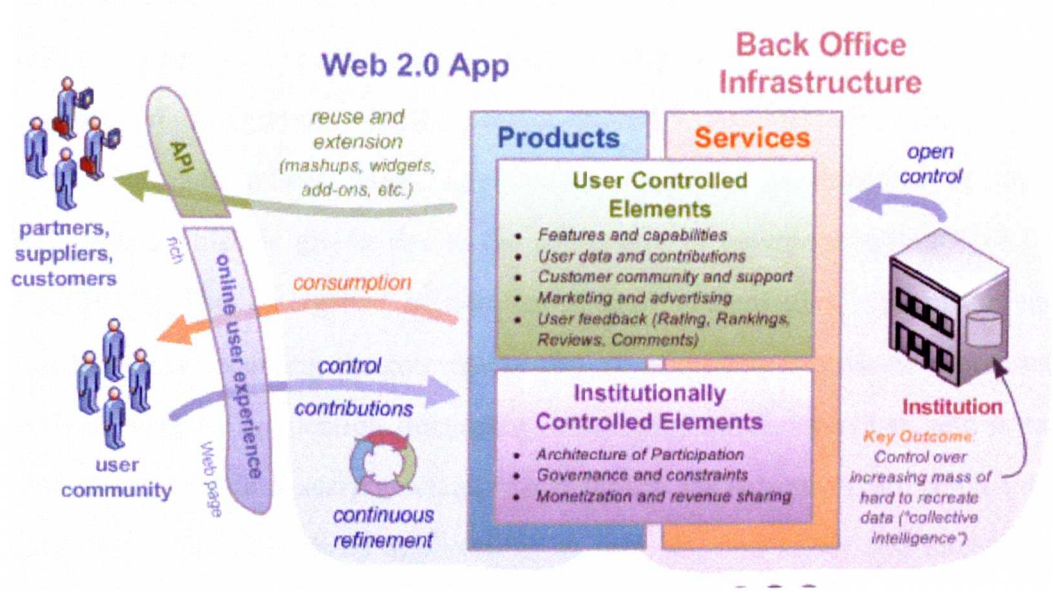


Figure 2.2: Web 2.0 product development (Hinchcliffe, 2007)

Figure 2.2 illustrates a typical Web 2.0 system development scenario along with the description of the user and institutional controlled elements. Institutions build up the functionalities in the form of web services and then Web 2.0 systems (or Web 2.0 products) are constructed using the power of these services. This is an approach to building new applications from existing functionalities. Web 2.0 systems also allow access to their functionalities in the form of API to other organizations acting as partners for the purpose of sharing functionalities. End-users play the role of consumer along with contributor in Web 2.0 scenario and contribute by supplying constant feedback for the improvement of the system (Dorn, 2009) and (Hsinchun, 2010).

2.3 Enterprise application architecture

Rapid advances in industrial information integration methods have spurred tremendous growth in the use of enterprise systems. Consequently, a variety of techniques have been used for probing enterprise systems. These techniques include business process management, workflow management, Enterprise Application Integration (EAI), Service-Oriented Architecture (SOA), grid computing, and others. Many applications require a combination of these techniques, which is giving rise to the emergence of enterprise systems (Li Da Xu, 2011). The emergence of service-oriented technologies, dynamism and flexibility are becoming the core characteristics of modern e-business processes, such as business application integration, distributed auction services, and order processing. Within a service-oriented architecture (SOA), an organization may encapsulate and publish its applications as services, and select and interact at runtime with the services provided by other organizations. However, such dynamic interactions at runtime raise immediate problems of security, trust, and dependability. Until these problems are addressed and solved satisfactorily, the potential of automatic e-business processes will be severely restricted. In a dynamic and distributed environment, it is often difficult for a complex business process to follow a static business specification. The execution order of its activities at runtime is usually unpredictable, and on some occasions, the actual execution of a process can be "one-of-a-kind" (Jie Xu, 2012).

According to (Choi, 2009) the emerging e-business practices require integration of core interoperable business processes to be reflected in e-business standards. Unlike traditional standards efforts that focused on IT infrastructure (e.g., data network communications protocols, physical interfaces, data format, etc.), the more recent trends and expectations in setting standards for e-business has been

to focus on interoperability across business functions and vertical/horizontal collaborations. In the field of e-commerce/e-business, such interoperability may be viewed from both business operations and IT infrastructure perspectives highlighting:

- Business aspects such as business data and information, business conventions, agreements, and rules among organizations and
- Technical concerns such as protocols and messaging architectures necessary to support business process execution at the transactional level.

In the research of (Zhao, 2008) e-Business implementation process is characterized with three dimensions and six constructs relating to e-business strategy implementation as illustrated in Figure 2.3, the six constructs are strategic initiative, information systems, partner e-readiness, IT human resources, information sharing capabilities (ISCs), and collaborative process capabilities (CPCs). This research describes and recognizes the effect of strategic initiative on deploying and utilizing IT related resources for creating e-business capabilities and the causal relationships. This leads to a better understanding of the underlying mechanisms in the e-business implementation process of the firm. In the model, ISCs and CPCs are conceptualized as new types of e-business capabilities to improve organizational performance. They are viewed as significant indicators for successful e-business strategic implementation.

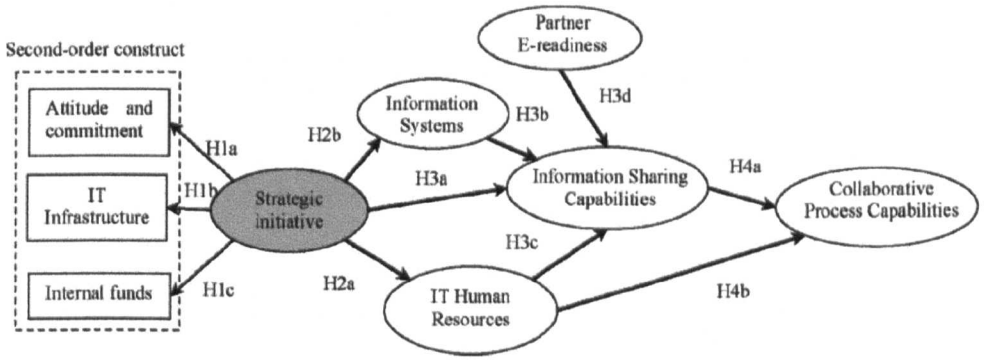


Figure 2.3: e-Business implementation process (Zhao, 2008)

2.3.1 E-business SOA-based layer Architecture

Juan (2010) provided a framework for the basic e-business system based on SOA. The whole proposed model from bottom to top is composed of 5 basic layers: network, logistics resources, services support, service integration and e-business, as shown in Figure 2.4.

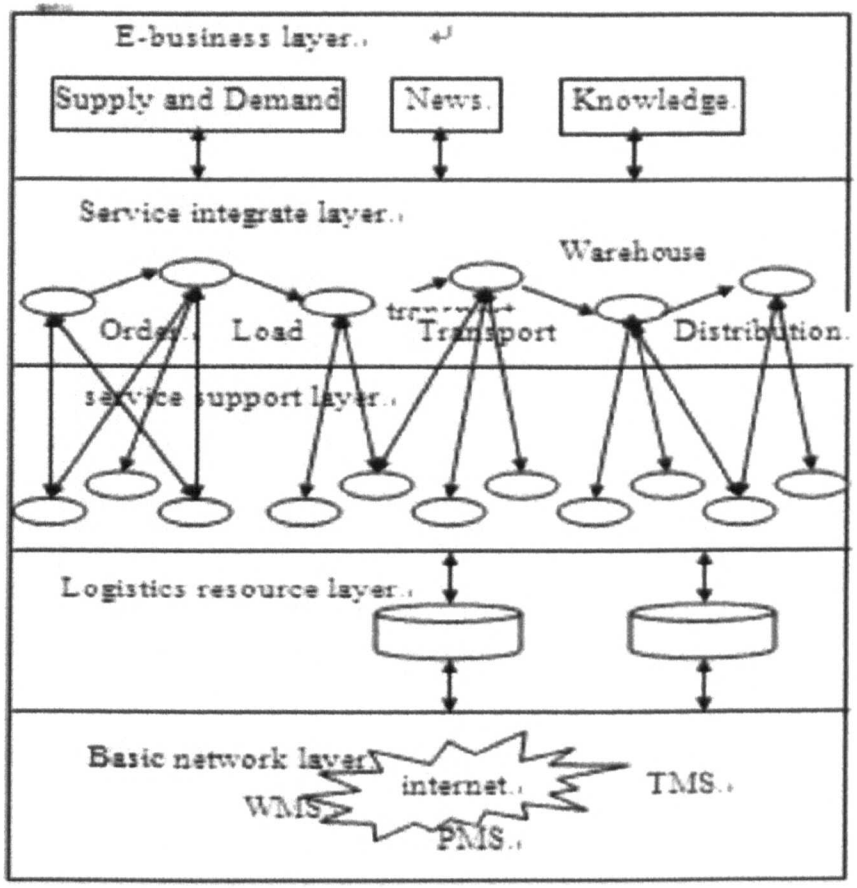


Figure 2.4: A framework for the e-business system based on SOA (Juan, 2010)

The e-business layer includes intranet portal, extranet portal and Internet portal. This layer provides information of supply and demand, logistics news, logistics knowledge, and online logistics business consultation etc. The e-business layer is a unified system interface and directly interacts with the user (including industrial and commercial enterprises, transportation company warehousing company, distribution, circulation and processing, etc) (Juan, 2010).

2.3.2 E-business SOA-based XML Schema Architecture

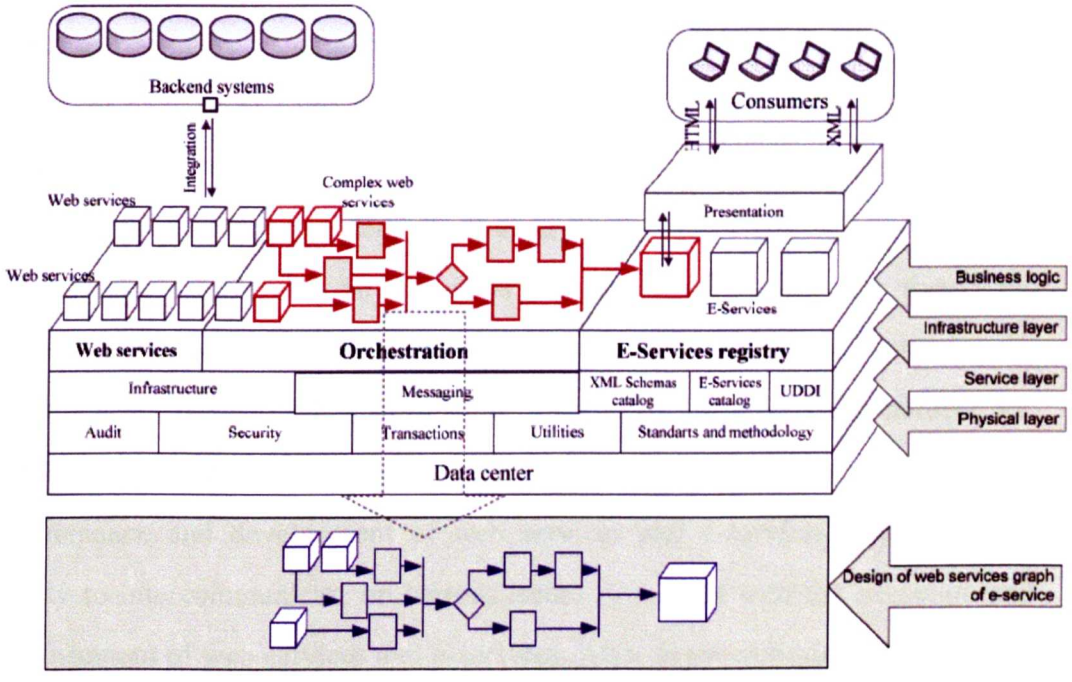


Figure 2.5: e-business system architecture (Edzus, 2010)

E-services are incorporated; (Edzus, 2010) proposed e-business architecture in a number of parts. It includes all the components, conditions and mutual links required to design e-services. Figure.2.5 shown how the systems and system components used in the service are combined into unified e–service system architecture (Edzus, 2010). For every data object that is required in the implementation of an e–service, an XML schemas catalogue has to be developed. Data called from the relevant functional system is done by means of web services. When web service calls are performed then metadata that describes the request is sent. In addition, any information that is required for audit trails is sent together with the metadata. Web services can be distinguished between two groups: simple and complex. Complex web services are logical combinations of

several simple web services that result from process integration requirements; a combination may comprise simple services of one or several independent functional systems. Complex web services can be executed by using a BPEL processor. A BPEL processor is used as the orchestration (integration) environment for the e-service's web services. Portals, one-stop agency applications etc ensure the delivery of e-services to users. E-service entry forms, stop points, information on payments and execution results are transferred through HTML or HML pages that can be used in the portal to implement the service using the XSLT transformation. Web service and e-service holders, i.e. institution specialists and system administrators who are responsible for the maintenance and development of web services and e-services, must have an ability to intercommunicate on various issues connected with the execution and advancement of web services and e-services. Also, asynchronous e-services have to be executed. Messaging systems are designated for this purpose. The messaging system enables working with text messages and work tasks (Edzus, 2010).

2.3.3 Service Oriented Enterprise Architecture

The service-oriented enterprise architecture (SOEA) is a development and operational architecture for enterprise integration in a service oriented computing environment (Zeng, 2009). SOEA provides an integrated development and execution environment based on SOA and model-driven architecture (MDA) as illustrated in Figure 2.6.

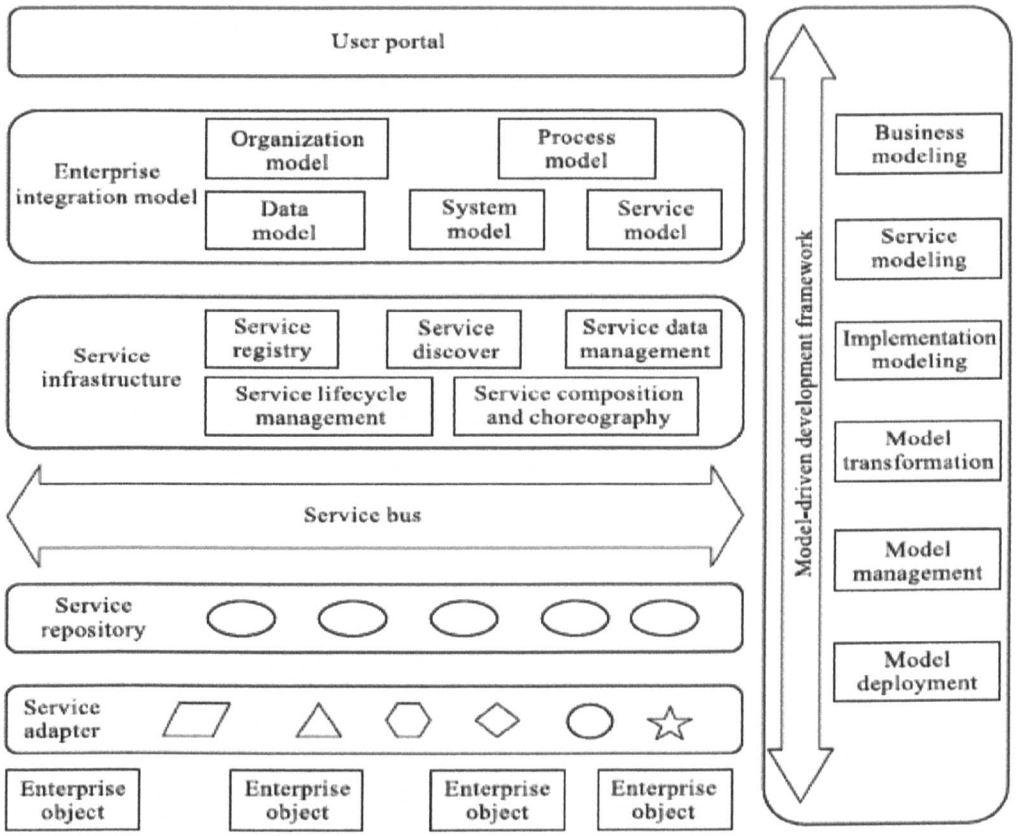


Figure 2.6: Service oriented enterprise architecture (Zeng, 2009)

Business processes supported by service oriented enterprise architecture (SOEA) include not only all traditional process elements, but also networked services (NS). Thus, a service-oriented business process (SOBP) may comprise the following three types of activities:

- The manual activity means an activity implemented by human beings. People can be helped by resources including enterprise applications and services.
- The network services activity represents an activity implemented by the NS which is normally called a service.

- If an activity is implemented by an enterprise application system that is not an NS, then the activity is an application-system-activity (Sen, 2009).

2.3.4 SOCRADES Integration Architecture

The research of Dominique (2010) proposes a process and a suitable system architecture that enables developers and business process designers to query, select, and use running instances of real-world services (i.e., services running on physical devices) dynamically, or even deploy new ones on-demand, all in the context of composite, real-world business applications. The proposed model is illustrated in Figure 2.7.

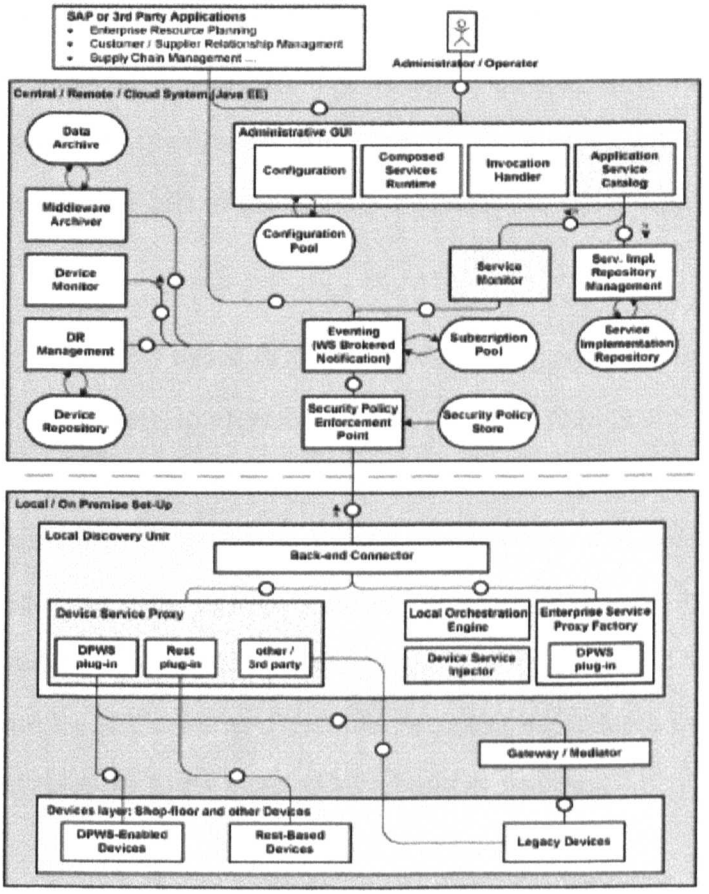


Figure 2.7: The SOCRADES integrated architecture (Dominique, 2010)

The process described in Dominique (2010) has been developed and implemented as part of the SOCRADES Integration Architecture (SIA) (Souza, 2008), (Karnouskos, 2007), (Spiess, 2009), which is depicted in Figure 2.7. The role of SIA is to enable the ubiquitous integration of real world services running on embedded devices with enterprise services. Web service standards constitute the de facto communication method used by the components of enterprise-level applications, and for this reason SIA is fully based on them. In this manner, business applications can access near real-time data from a wide range of networked devices through a high-level, abstract interface based on web services. Furthermore, the SIA also supports RESTful services in order to be able to communicate with many emerging Web 2.0 services. This enables any networked device that is connected to the SIA to participate directly in business processes while requiring neither the process modeller, nor the process execution engine to know about the exact details of the underlying hardware (Dominique, 2010).

2.3.5 MerchantOS

MerchantOS is a Web 2.0 based POS system developed by a US-based company. The system is completely Internet based and does not require the installation of any additional software. Interested companies (having one or more POS) can get access to the MerchantOS by paying a small monthly fee. The cost of the system is customizable depending on the features required. It ranges from $29.95 to $99.95 per month. The system offers the transformation of manual POS, Inventory Control and Customer Relations business procedures into e-business. Figure 2.8 illustrates the login page of MerchantOS system.

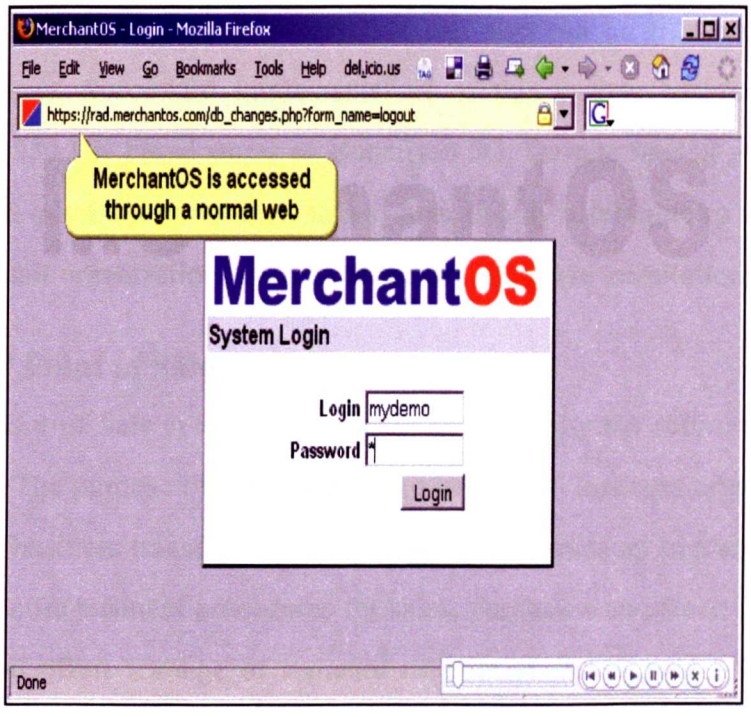


Figure 2.8: MerchantOS POS System (MerchantOS, 2007)

Complementary to all offered features, the system requires installation of related hardware such as receipt printer, cash drawer, label printer, barcode scanner and credit card reader. Moreover, MerchantOS only offers standard business procedures and user interface to run POS system and it does not offer any customization facilities, such as customization of standard business logic into company-specific business logic, or customization of standard user interfaces into company-specific user interfaces.

2.3.6 Retail-J

Retail-J is another Web 2.0 based integrated suite of in-store and central applications for medium and large size retail companies by Torex Retail. Retail-J a complete retail solution offering features ranging from multi-version POS to a complete back-office system. The POS system offered by Retail-J is not only

capable of running on web browsers, but can also run on tills and hand held mobile devices. Successful implementation of this system has recently been achieved in 250 UK based stores of Hutchison 3G. Despite having good features such as cross-platform portability, this system is only affordable to medium and large size retail organizations and requires extra hardware installation.

2.3.7 PHP Point of Sale

The PHP Point of Sale system is the result of efforts by the software developer community. The purpose of this system is to help small business organizations in achieving e-business transformation. This system contains an implementation of industry-specific business procedures for sales, purchases and stock management activities and offers a range of standard user interfaces. Figure 2.9 presents a view of a sales screen from the demo version of the system.

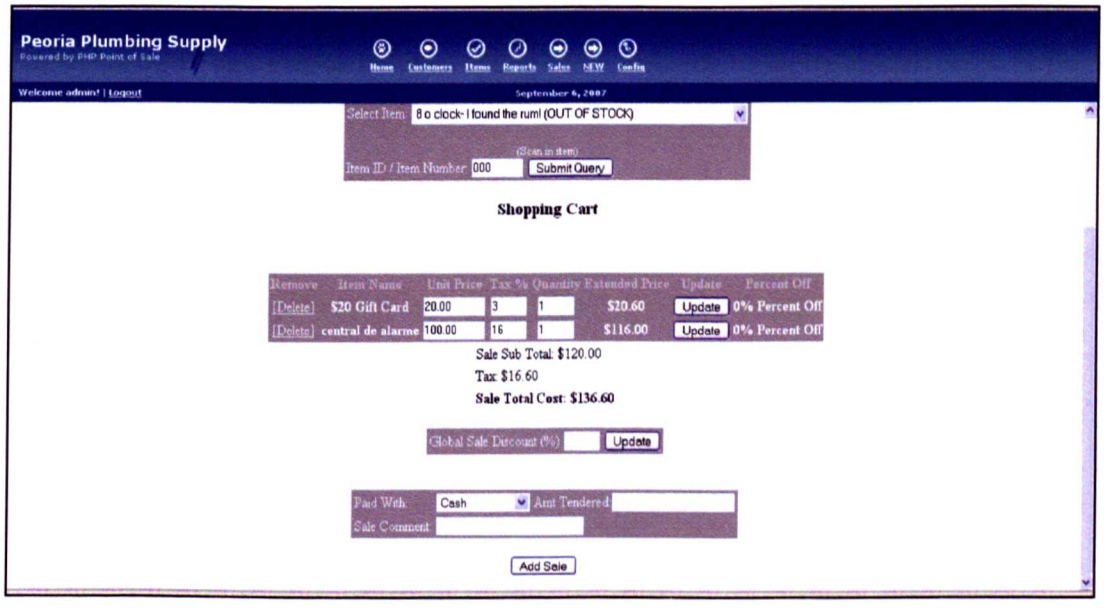


Figure 2.9 : PHP Point of Sale System (PHPPointOfSale, 2007)

Despite being an open-source project, this POS system is not usable for small retailers who require business logic and/or user interface customization. If a retail company decided to use this system they would have to hire an IT professional to undertake the customization needs of the business. Most importantly, no technical support and API is available from sourceforge; therefore the upgrade and maintenance responsibility of the system relies solely on the organisation using it.

2.3.8 ZohoCRM

Zoho is one of the prominent Web 2.0 Companies which is progressively making their contribution to the Web 2.0 field. They have transformed various desktop applications into Web 2.0 systems. From a large array of their Web 2.0 products, they offer ZohoCRM, which is an affordable on-demand customer relationship management system with integrated POS. Any individual or company that runs a retail business can access the system, and is allowed to create up to three free user accounts. Zoho start charging retailers from fourth user account and the fee is as little as USD 12 per user per month.

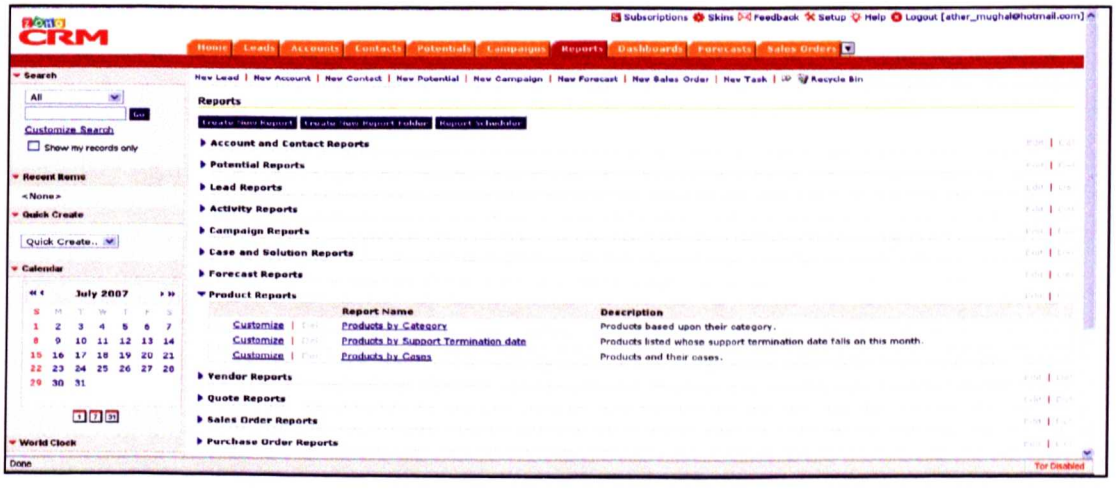


Figure 2.10 : ZohoCRM System (ZohoCRM, 2007)

In addition to standard features, ZohoCRM also offers limited customization of various business reports to its client companies (see Figure 2.10). Despite offering valuable features such as limited free access and customized business reporting, ZohoCRM does not facilitate on-demand customization of its business logic and other user interfaces.

2.3.9 A brief analysis about enterprise applications

The thorough review of the current Web 2.0 retail systems such as MerchantOS, Retail-J, PHP Point of Sale and ZohoCRM etc that are built upon the existing SOAW2 model framework has shown the lack of fundamental functionalities that are desirable in an e-business framework for SME's, such as on-demand customization of business logic and user interfaces. A system like ZohoCRM (zohoCRM, 2007) has shown some capabilities of user-interface customization but it is only limited to the customization of business reports. One of the important observations is that, none of current Web2.0 retail systems are capable of providing on-demand customization of business logic due to the fact that applied SOAW2 model framework does not support explicit request management. The current SOAW2 model framework directly exposes user interfaces to business services that reside within the resource container. This direct exposure results in the form of user interfaces that contain concrete business services mapping instructions. Due to fundamental shortcoming of the model framework, developed systems do not provide on-demand customization of business logic at run-time. Secondly, the model framework does not provide any extra layer for dealing with this customized business logic. Therefore, an investigation is required to propose a new Web 2.0 model framework that will overcome these fundamental shortcomings of SOAW2.

2.4 Web 2.0 based SOA models

Web 2.0 is a useful concept that is based on the idea of developing software as service by considering the Internet as a platform, and utilizing the power of user contribution towards its improvement. In other words, it is a concept of building server-side software as opposed to conventional client-side software where end-user feedback plays a vital role in its evolution.

A research conducted by (Abbas, & Bendiab, 2007) proposed the model framework for defining Web 2.0 components and their relationships. This proposed model is called SOAW2. They apply the SOA approach to underline the important components of Web 2.0 and SOA technologies in 5 core and 3 managing and protection support layers, as illustrated in Figure 2.11.

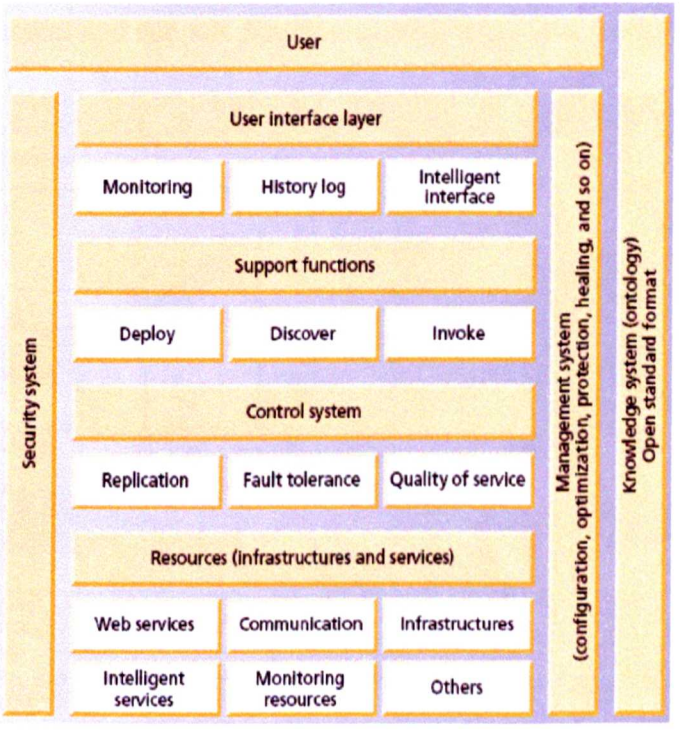


Figure 2.11: SOAW2 model framework by (Omar, Abbas, & Bendiab, 2007)

Web 2.0 systems are incomplete without the user, and hence their model framework includes a 'User' as a separate layer. The Resources layer in SOAW2 acts as a container for core services such as financial services, stock services etc. These services play a part in building additional services. The Control system layer represents a placeholder for a model manager which manages and controls the services sitting in resources layer. The Support function layer contains standard SOA functions for service providers to deploy services. The Discover and invoke functions are reserved for the user interface layer. The user interface layer contains set of user interfaces for end-users and acts as a gateway for users, to enable access to services. The User layer represents an end-user (i.e. a consumer) who interacts with the system by means of the user interface layer. The remaining three managing and protection support layers encapsulate the overall management and support function capabilities of the model framework.

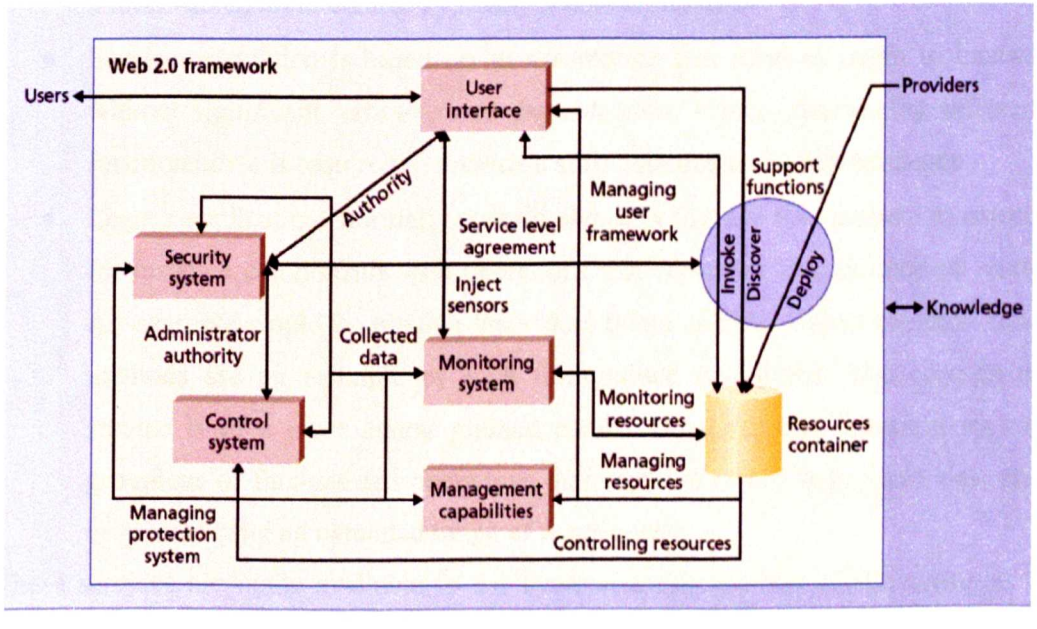


Figure 2.12: SOAW2 in action (Omar, Abbas, & Bendiab, 2007)

Figure 2.12 illustrates the collaboration between different components of the SOAW2 model framework while in action. It is worth mentioning here that SOAW2 is not a new model framework, rather, it is an outcome of research activity that analyzed in depth the internal working models of existing Web 2.0 systems.

The Service Oriented Architecture (SOA) is becoming a mainstream approach for designing and integrating enterprise applications. The research conducted by (Sheikh, Aboalsamh, 2011) proposed a mechanism to convert the old systems in to SOA systems. The proposed solution applies the following principles of service-orientation:

- It is impossible to isolate SOA services directly while business logic is bundled with presentation logic. Therefore, an appropriate restructuring and reorganization of the code is required so that business functions are isolated as candidate services or service components. This technological dimension of reengineering towards SOA represents an architectural transformation towards a multi-tiered architecture. Typically 3 tiers are used.
- Service-orientation is based on an assumption that services ought to interact without significant, cross-service dependencies. Hence, rearranging different functionalities is required to provide a sufficient degree of independence.
- Legacy applications normally contain elements that are fine-grained in nature, for instance components with operations that represent logical units of work. An example would be reading individual items of data. Object oriented class methods are an example of such fine-grained operations. The concept of service is of a more coarse-grained nature. SOA services represent logical groupings of fine-grained operations that work on top of larger data sets, and in general offer an extended range of functionality.

These services are made available to the Enterprise Service Bus (ESB) acting as a mediation, service and protocol virtualization. This layer is accessed by the top level business process management and presentation applications / systems. This research did not address problems such as missing data packets between the business layer and the user interface or load balancing of the requests etc.

In the last decade or so, we have witnessed the software industry moving towards service-oriented architecture based technologies. Especially in the business software domain, complex applications based on the composition and collaboration among diverse services has appeared. According to the research conducted by Dominique (2010) the service-based information systems blur the border between the physical and virtual worlds, providing a fertile ground for a new breed of "real-world" applications.

The advent of Service Oriented Architectures (SOA), and the automation of business processes can be aided by service composition (Erl, 2005) which is dynamic and creates opportunities for runtime management. A single business process can encompass many different functional components such as orchestration engines, legacy middleware, application servers, data transformation hubs, and web service facades. Governance of such complex business processes requires a monitoring system as a starting point for the enforcement of relevant management decisions. The motivation for monitoring of frameworks, middleware solutions, engines, and databases has been addressed in many studies (e.g., (Wang, 2009), (Simmonds, 2009), (Pistore, 2004), (Liu, 2007), (Yuan, 2008), including, among others, performance evaluation, security evaluation, testing conformance with requirements, ensuring stability, and detecting anomalies. This motivation is especially important in distributed, loosely coupled and dynamic SOA systems, where the required level of Quality of Experience (QoE) and Quality of Service (QoS) cannot be ensured without sophisticated monitoring of interactions between consumers and providers. Ongoing studies related to SOA system monitoring focus on specific elements, i.e., orchestration engines (Baresi, 2005), (Barbon, 2006), (Moser, 2008), (Wetzstein, 2009) and web services (Wang, 2009), (Simmonds, 2009). However, this is not enough for end-to-end monitoring of SOA systems, which often span multiple functional elements and automate cross-organizational business processes.

Open Service Process Platform 2.0 is a central SOA infrastructure including several extensions. This platform can be used as the basis for the development of further extensions.

The unique features of the platform are:

- Orchestration and execution of processes in an easy way.
- Arbitrary extensibility with regard to simple specialization for various domains.
- Central infrastructure within an organization.
- Full accessibility through Web 2.0 technologies.

Firstly, with the help of this platform, it should be possible to both orchestrate and execute processes. Secondly, access to the platform should be available from everywhere, so that, for example, the current state of the process execution can be monitored, necessary interaction with the processes can be performed, process definitions among users can be shared and users are able to design processes in a collaborative way. Thirdly, the platform should be easy to extend with the help of a plug-in concept to make it possible to customize it for different domains. Fourthly, the platform should take on the role of the central process platform within an organizational unit, i.e., we require repositories for services and processes as well as a rights management system.

The research of (Marek, 2012) presents an end-to-end solution on SOA systems by focusing on the Integration Layer, part of the SOA Solution Stack. The proposed architecture provides detailed architectural definition of an SOA across nine layers, which aim to reinforce business value. The stack defines five horizontal layers: Operational Systems, Service Components, Services, Business Process, and Consumers. The horizontal layers are cut across by five vertical layers: Integration, Quality of Service, Information architecture, Governance, and policies. Integration layer focuses on bringing together layers of Service Components, Services, and Business Process. This model points to the Enterprise Service Bus (ESB) as the most appropriate solution for implementing the Integration Layer, whose purpose is mediation, routing, and transporting service requests from service consumers to correct service providers.

The Service-oriented architecture is an architectural paradigm for building software applications from a number of loosely coupled distributed services (Baker, 2012). This paradigm has seen wide spread adoption through the web services approach,

which has a suite of simple standards (e.g. XML, WSDL, and SOAP) to facilitate interoperability.

2.5 Request management principles

One of the major deficiencies of the SOAW2 model framework is the absence of a request management mechanism. SOA (Nickull, 2007) allows enterprises to centralize computer-based services and offer those services over a network. On the basis of a published interface, the service can be used or a platform-independent basis within and outside the enterprise. The concept of a SOA is often realized by web services (WS). The interface of WS is usually described by WSDL (Web Service Description Language) in accordance with SOAP (Simple Object Access Protocol). Both standards, SOAP and WSDL, are based on XML.

Interaction with web service requests and responses acts as events within the environment (human users or other systems). Requests' function calls WS which will supply appropriate responses. In spite of its functional correctness, a request may be useless if it is not delivered "in time", e.g., a request for a stock price is of no interest if it is not delivered before the next change (Nickull, 2007). Taking such time constraints into account requires time monitoring from request to response; in other words, we need a real-time service-oriented architecture (RTSOA). Time can be measured either server-side or client-side. The time difference between client and server time primarily depends on SOAP calls which entail intense XML parsing. Also the size of a response plays an important role: A search function, triggered by a single query, can return huge amounts of entries. To this end, subsequent sections contain discussion on the classification of request management techniques followed by a detailed review of relevant request management techniques. There are only two types of request management

technique currently available: conventional, and request brokerage. The following paragraphs elaborate further on this request management technique.

2.5.1 Conventional technique

The conventional request management technique is a technique that is commonly used in traditional web-based systems. In this conventional technique, request management is automatically performed by the application server. For example, in an ordering system scenario, on submission of a sales order form by a user, a request goes back to the application server. On arrival of the request, the application server automatically redirects it to the designated sales order service for processing. At the end of processing, the service directs control back to the application server by specifying the appropriate response page address (e.g. confirmation or error page). Finally the application server loads the response page and sends it to the user. The biggest disadvantages of this technique are its limited usage and incapacity of handling complex scenarios, such as a generic business login to handle the "make sale" process etc. The request brokering technique has proven its successful implementation in many complex scenarios, such as in (Hitachi, 1999), (XiaoQin, 2004) and (Yerom, 2009) etc.

2.5.2 Request brokering technique

The concept of dynamic service binding through request brokering is an active research topic. This section presents findings of a thorough literature review of the latest research attempts made by different researches in the field of request brokerage. It also elaborates their respective drawbacks and associated issues. The primary focus of this effort is to find out any attempt which can resolve the request management problem, as discovered in SOAW2 model framework. If

not, then it could at least form the basis that justifies the need of a new request brokerage technique.

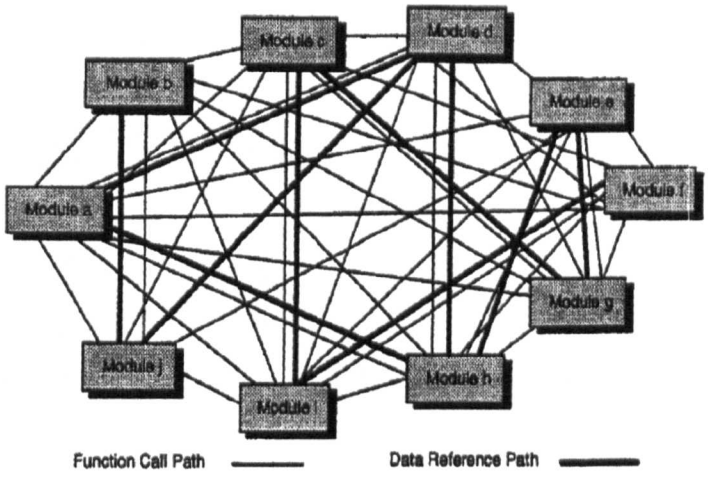


Figure 2.13: Modular dependency in a system (Olson & Batni, 1997)

What does request brokering replace? It is an important question that needs to be addressed. Figure 2.13 illustrates a modular system, in which different modules are connected to each other. The Figure shows an intensity of modular dependencies within a system. This modular dependency between different modules makes any system very complex thus limits system scalability and increase complexity. This modular dependency is very common in network based system and to reduce this modular dependency (Olson & Batni, 1997) proposed the concept of service brokerage architecture. Figure 2.14 illustrates their proposed request broker model.

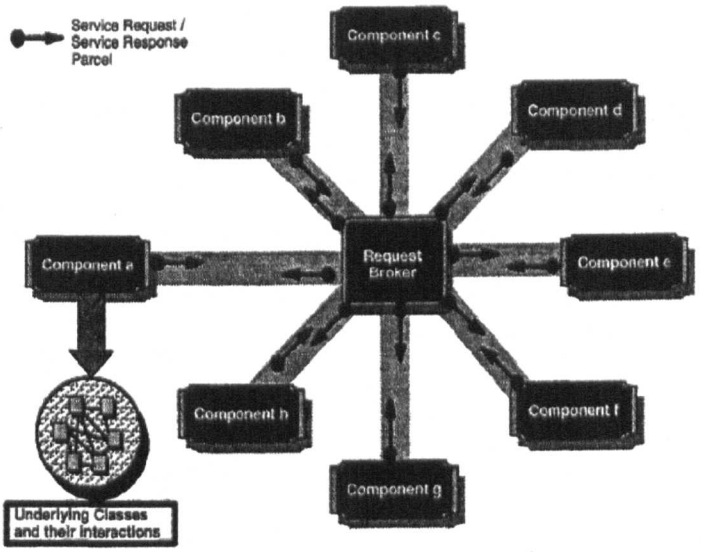


Figure 2.14: Request Broker Architecture as proposed by (Olson & Batni, 1997)

In their proposed model, they connect every component with a centrally placed request management component known as the request broker. Introduction of this request management component synthesizes the way different components communicate with each other.

This concept of request broker-based system architecture became the focus of interest for the research community and is still gaining popularity. Request brokerage in service-oriented architectures is an active research area. Research community explored the suitability of a request brokerage in SOA based systems, such as in (ACTS, 2000) and have proposed some stable models in which a request can be dispatched to the most suitable service(s), transparent to the user, with the help of request brokers. Literature is scarce on the topic of dispatching back a response from the service to the most suitable user interface. In simple terms, current request broker architectures support action management but not view management.

(Howard & Kerschberg, 2004) proposed a complete framework called KDSWS (Knowledge-based Dynamic Semantic Web Services). Their proposed framework approach is mainly based on a multi-agent system in which web services have been converted into semantic web service. KDSWS's brokerage technique is based on four agents, namely broker agent, classification agent, discovery agent and selection agent. The framework applies a rule-based approach to facilitate the run-time generation of the dynamic profile. Using this dynamic profile brokers discover, negotiate and finally bind the service that best meets the user's needs. In contrast to offered features, this proposed framework does not facilitate a response mapping back to a user-defined interface. In addition, the proposed model is not applicable in situations where a request demands execution of more than one service or requires a partial collaboration of different services in order to fulfil user needs.

In other research (Beck, Konana, Liu & Mok, 1999) proposed an idea of a next generation electronic brokerage for performing active and real-time functionalities. Since every standard brokerage system follows a set of activities to bind services, the execution of these activities sometimes becomes very costly especially when a user request is of a complex nature involving time as a major factor. The proposed brokerage model in this research deals with the timeliness factor by involving an active database in the brokerage scenario. This model allows the user to express complex preferences in the form of Event, Condition and Action (ECA) rules. Every request gets forwarded to the request brokers along with these ECA rules. Brokers continuously monitor the request and if a specified event occurs, and the condition (which is a predicate or database query) becomes true, they execute the specified action(s) in a timely manner. These actions, triggers the pre-complied routines of attached pro-active databases.

These pro-active databases run as a part of brokerage system and support pro-active transactional triggering. In nutshell, the execution of requested services is achieved in a timely manner by involving databases of a pro-active nature. This brokerage model contains architectural deficiencies such as involvement of pro-active databases making the model implementation dependent on databases. Moreover, no concerns have been expressed on mapping responses back to a view that is acceptable to user.

The request broker model proposed by (XiaoQin, LinPeng, Lin, & Minglu, 2004) is a concept of an agent based web-services platform. This proposed model has been successfully implemented on Tele-portal solution for Shanghai Information Science and Technology. The scope of the proposed brokerage model is to handle e-commerce operations by providing assistance to consumers and operators. Automated assistance (by request brokers) helps consumers in finding feasible offers and at the same time assists operator in marketing different offers to the consumers. The request brokers' model proposed in this research is actually an extension of the Architecture for Information Brokerage Service models proposed in (ACTS, 2000).

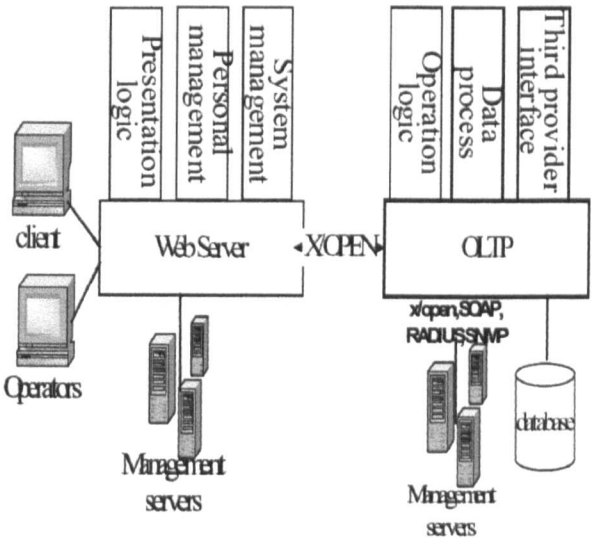


Figure 2.15: Agent based web services platform for Tele-portal (XiaoQin, LinPeng, Lin, & Minglu, 2004)

Figure 2.15 illustrates the Tele-portal system that was built by applying the proposed model. The complete system is physically divided into two hardware units – Web Server and OLTP (On Line Transaction Platform). The portal core exists on a web server and is composed of three modules, namely presentation module, logic module and entity module. The portal reserve proxy exists in front of a web server where client and operators communicates. This portal reserve proxy houses the multi-agents (i.e. request brokers). These multi-agents break down the client's request on arrival. This is a part of a divide-and-conquer technique used to solve complex and distributed requests. Despite having a real-world implementation of the proposed model, the model does not support the customization of business logic and user interfaces. Presentation and operation logic components of the system only contain a default set of user interfaces and a default set of business logic, respectively.

A CORBA-based mediation platform design for efficient brokerage is proposed in (Koerner, et al., 1999). The design is primarily based on the concept of a complete mediation platform presented in (Tothezan, Athanassiou, Alzon, & Karetsos, 1997). The proposed model also complies with TINA and OMG specifications. The design beauty of this model is in its dynamic profile and profile rule management techniques. Unlike (Howard & Kerschberg, 2004), the model has an applied profile management philosophy on both user and service provider ends.

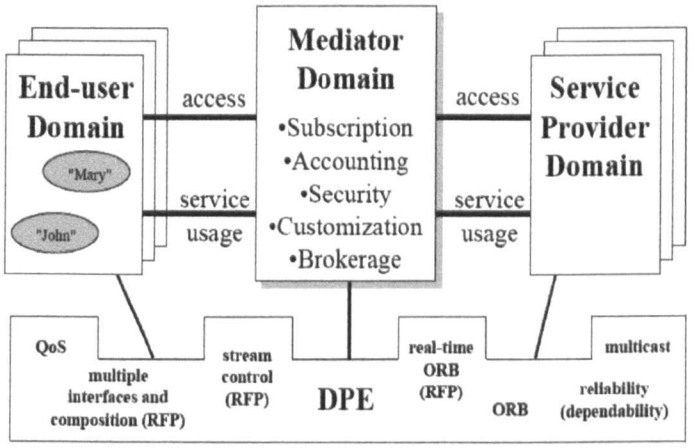


Figure 2.16: Service mediation in ADSS (Koerner, et al., 1999)

The research project was collaboration between Hitachi, Ltd. and GMD FOCUS to develop a new ADSS architecture to apply the ADSS concept on an application level; especially in the context of electronic commerce and social information systems. Figure 2.16 depicts a visual design of service mediation in ADSS. The ADSS model classified business in three major business roles namely mediator (which mediates the services between parties by maintaining a high level of service agreement), end-user (a party who acquires and consumes the services according to agreement with the mediator) and service provider (a party

who publishes services information in mediator domain to be consumed by the end-user). Because of the mediation domain in the middle, end-users have flexibility in choosing the services they want without worrying about the low level details. Similarly, service providers get the flexibility of offering services without going into technical details such as publishing, discovery and binding of the services.

The proposed ADSS model supports profile management techniques. It spans the entire service provisioning process including service brokerage, subscription, access sessions and service session. Service providers publish their service related information in the form of service templates for type information description. End users have the choice of subscription to these services by using these service templates, and can also customize services by specifying the features they want. Despite a sophisticated model for service brokerage, the proposed model does not support the customization of output data (i.e. user interface customizations). Moreover, service usage scenarios presented in this paper are based on the assumption that complete user requests will be fulfilled by only one service, hence, the proposed model is not appropriate in scenarios where service composition is required to fulfil one user request.

Research conducted by (Zhao & Tong, 2007) has proposed a service composition model: 'A Dynamic Service Composition Model Based on Constraints'. This model is not a request broker based model, but is capable of handling complex situations where a user request has a very complex execution nature such as requirement of partial (or full) execution of more than one service. Figure 2.17 illustrates the proposed model. A domain expert initially sets up the Domain Ontology for a domain and put it into the registry. This domain ontology defines

common information types, the quality indexes for this domain and a set of abstract workflows. A user interacts with the system via the user's portal, then the user selects an abstract workflow and specifies requirements as a set of constraints. The Configuration Engine receives the user request and converts it into a concrete workflow using domain ontology. This concrete workflow contains every detail of service(s) required to perform the request. The configuration engine then binds the matching services in the execution order using the registry and passes it to workflow engine for execution as shown in Figure 2.17.

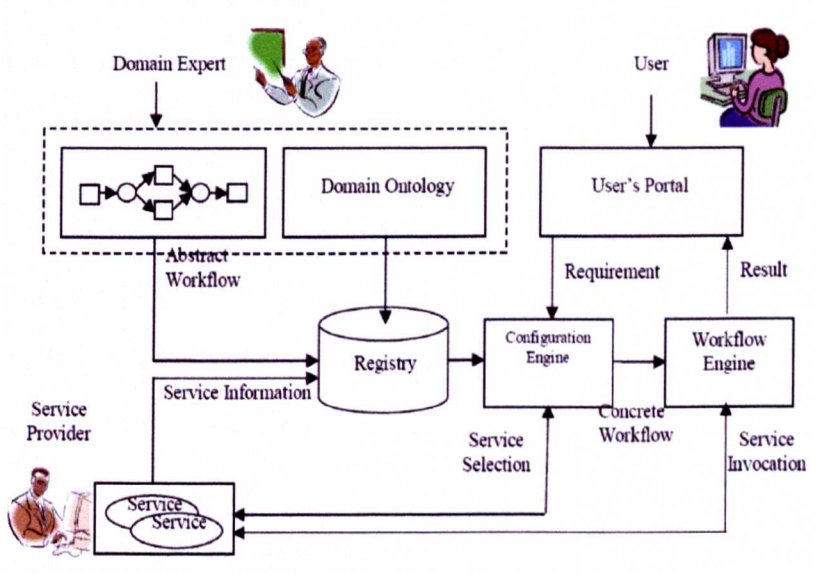


Figure 2.17: Service composition model (Zhao & Tong, 2007)

Unlike the other models presented, this model leaves a selection of abstract work flow to user. This technique could be beneficial where a user is an expert user having a sound knowledge of the business domain, but in case of a novice user this technique could have drastic effects such as execution of wrong business

logic. Secondly, this model has presented no explanation on how response data will route back to user.

A review of request management techniques above has shown that only the request brokering technique has proven its strategic importance and successful application in complex scenarios. Therefore it can be considered as a candidate for fundamental changes in the SOAW2 model framework. It has also been observed that existing request broker techniques are more service-centric than view-centric, they are primarily focused on routing a user request to best matching destination service but are incapable of routing service output back to best matching destination user interface. Hence, existing request broker technique needs to be improved and then integrated in a newly proposed model framework as a request management component.

2.6 Conclusion

The research studies introduced above were conducted over the last decade. During the review all aspects of the SOAW2 model framework were discussed in detail along with its application. None of the Web 2.0 retail systems built upon the SOAW2 model framework exhibit characteristics that are required in the e-business framework such as on-demand customization of business logic and user interfaces. Amongst other factors, one major factor of this shortcoming is the absence of a request management component. Therefore, the proposal of new Web 2.0 model framework is inevitable and is a key motivation of this project.

Moreover, review of popular request management techniques has shown that only the request brokering technique has proven its strategic importance and

successful application in complex scenarios as required in an e-business framework. It can therefore be considered as a candidate for fundamental changes in the SOAW2 model framework. It has also been observed that existing request broker techniques are more service-centric than view-centric, i.e. they are primarily focused on routing a user request to the best matching destination service, but are incapable of routing service output back to best matching destination user interface. Hence, existing request broker techniques need to be improved and then integrated in a newly proposed model framework as a request management component.

3. Conceptual Modelling of New Web 2.0 e-Business Framework

3.1 Introduction

Enterprise applications are increasingly being architected in a service-oriented architecture (SOA) style, in which modular components are composed to implement the business logic. The properties of such applications, such as the loose coupling among the modules is promoted as a way for an agile business to quickly adapt its processes to an ever changing landscape of opportunities, priorities, partners, and competitors. The proliferation of Web services standards in this area reflects the industry interest and demand for distributed enterprise applications that communicate with software services provided by vendors, clients, and partners.

3.2 SOAW2 model framework evaluation

The model framework is based on the Web 2.0 paradigm. Therefore, prior to proposing a new model framework, an evaluation of the existing SOAW2 model framework in this context is essential.

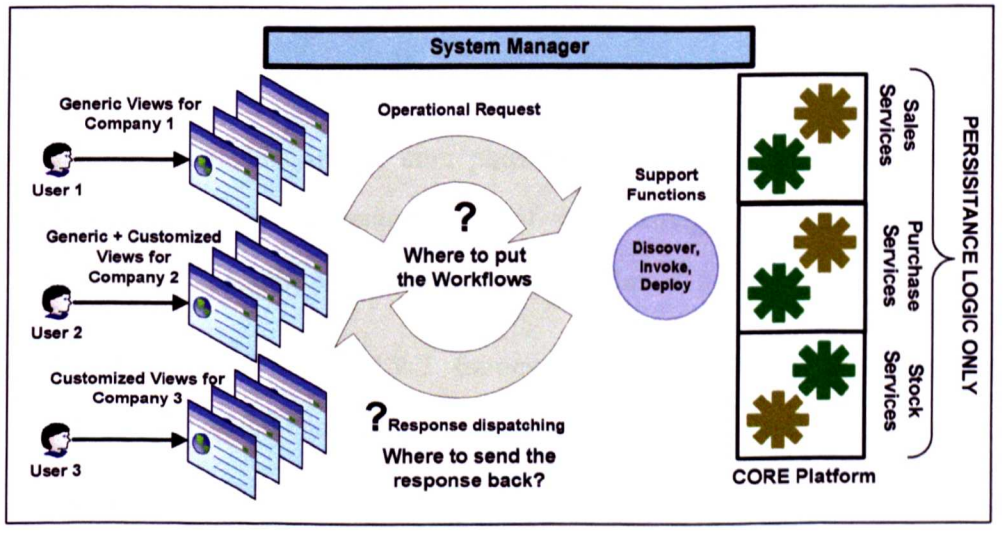


Figure 3.1: Application of SOAW2 model framework on problem

Figure 3.1 presents a visual representation of the problem solution with the application of the existing SOAW2 model framework. As discussed and concluded in the literature review, chapter 2, it was found that the SOAW2 framework does not support explicit request mediation. Due to this architectural deficiency, every company that runs its e-business on a system requires its own dedicated set of views with the names of required services embedded inside them. Figure 3.1 illustrates such limitations where the users of company 1 and company 2 are using replica copies of the same generic views, whereas the users of Company 3 have their own set of customized views. As a result, this system does not offer any sharing of user interfaces.

In addition, scalability of the system is a big concern. Every time a new company is registered on the system, the service provider has to allocate a dedicated set of views to this new company.

3.2.1 Concerns

The application of the SOAW2 framework in a given problem scenario has also raised the following important concerns.

- Services that exist in the core platform contain only persistence logic; they contain no implementation of workflows. The SOAW2 model framework does not provide a slot for workflows.
- Applications of the SOAW2 framework do not contain any response navigation logic; hence there is a question of where to send the response.
- Finally, due to the non-sharing nature of the model, if a generic view requires an update, then a solution provider company has to update each single matching generic view in every company's view set. Since this

requires more time and will increase the cost of administration, the solution provider company will recover this cost from retailers. How can the system then be classified as a shared and affordable system?

The SOAW2 model framework has no answers to these critical questions. Hence, it can be concluded that existing model frameworks are not sufficient to be applicable for generic e-business platforms and require fundamental changes.

3.3 Proposed web 2.0 model framework

This section describes the new Web 2.0 model framework called W2ASVB (Web 2.0 Architecture for Service and View Brokerage). The new model framework is based on SOAW2 and also contains two fundamental changes in the basic model framework. These fundamental changes include the introduction of an effective and intelligent request broker architecture, and the replacement of supporting functions with service adapters. Figure 3.2 illustrates the W2ASVB model framework. Justification of these two fundamental changes is given in the following sub-sections.

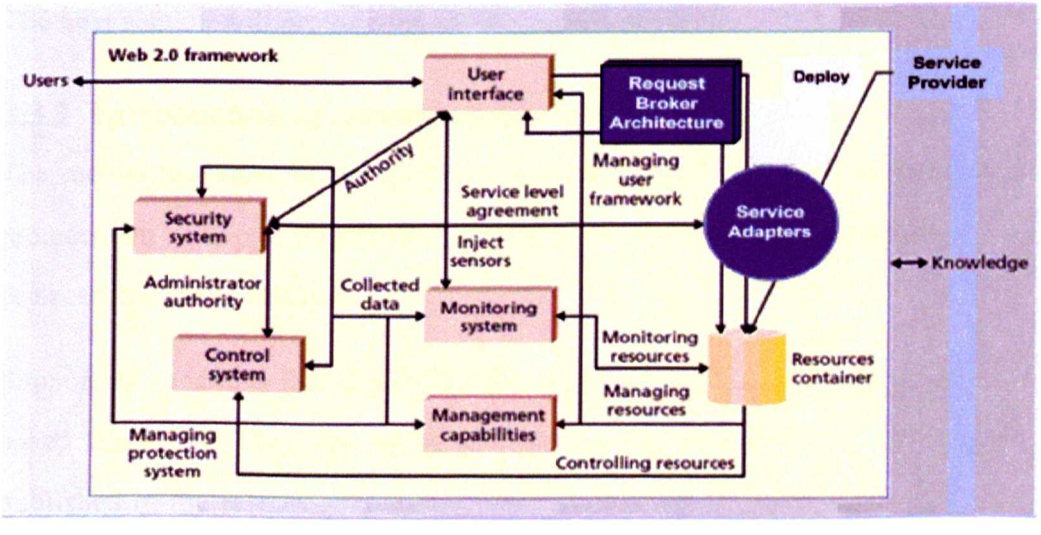


Figure 3.2: W2ASVB - A new proposed model framework

3.3.1 Introduction of the request broker architecture

By taking into consideration the shortcomings of the request management component in SOAW2 and the need for an effective and intelligent request brokerage mechanism to handle complex on-demand sharing and customization issues, a new request broker architecture is integrated between the user interface and the resource container, as shown in Figure 3.2. W2ASVB consists of a new request broker mechanism concept. This request broker concept improves on existing request brokers, which are responsible for action and view management inside a W2ASVB model framework and eventually provide on-demand request routing between user interfaces and services of core platform.

The knowledge layer of the model framework is shared between service providers and users (i.e. employees of retail companies). This knowledge layer is a typical requirement of Web 2.0 model frameworks and it is there to represent the architecture of participation. This means that new knowledge in the model will evolve through the experiences of both users and service providers. Now, it is worth asking the question, "how and where should the workflows be handled"? The next section will provide the answer to this question.

3.3.2 Introduction of service adapters

The second fundamental change to the proposed W2ASVB model framework is a replacement of supporting functions with service adapters. The reasons for this replacement are as follows:

Supporting functions like deploy, invoke and especially discovery are pure SOA based functions. They are useful in situations where one type of service is published in the resources container from more than one service provider and a user request needs to be resolved to the best matching service. In this case, the

discovery function provides help in resolving the best matching service within the resources container. Moreover, a pure SOA approach requires deployed services to comply with W3C standard specifications as described in (WC3, 2004). However, in W2ASVB, services that are deployed inside the resources container (i.e. the core platform) are implemented by applying a SOA based approach but they do not comply with W3C standards, as W3C standards services are deployed outside the core as a separate entity. Therefore, associated supporting functions are no longer needed.

The replacement of supporting functions in the new model framework is provided by service adapters. A service adapter is a new concept of "light weight" services and contains implementation of workflows. In the SOA tradition, services are relatively large, shared, intrinsically loosely coupled units of functionalities, and have no embedded calls to each other. Web Services are a version of SOA which run over the Internet and provide services to users transparent to their locations. These web services expose their services to the outside world through end-points written in WSDL and communicate with users using SOAP. There is still concern that if W2ASVB requires sharing of generic workflows among different retailers then why can generic workflows not be modelled as web services? The simple answer to this is that, in W2ASVB, workflows are actually sequential calls to the services of the core platform. Therefore, if workflows are modelled as web services, it would be a violation of the principles of loose-coupling and no embedded calls within SOA.

To overcome this limitation and to avoid any conflict with the SOA principles, workflows are modelled as service adapters. After receiving a request from a user through the user interface, these adapters are connected with the core platform to

execute the workflow modelled inside them. All the workflows (including generic and customized) are modelled as service adapters. Generic workflows are executed by the generic service adapters, whereas customized workflows are executed by customized service adapters.

3.3.3 Proposed work flow model

A user request for data processing will be divided into two parts – a data and an action part. The data part contains the *data* that requires processing, whereas the action part describes the required operation on the given data. Request brokers are a key concept in the proposed W2ASVB model framework. They are semi-autonomous objects which are completely equipped with all the capabilities to handle action and view management for user requests. In other words they can be called middle agents that provide a mediation mechanism. They have been classified as semi-autonomous because they can complete the user's request without any assistance from other brokers but are bound to be controlled by managing authority. This managing authority is called the system manager and is summarised in Figure 3.3.

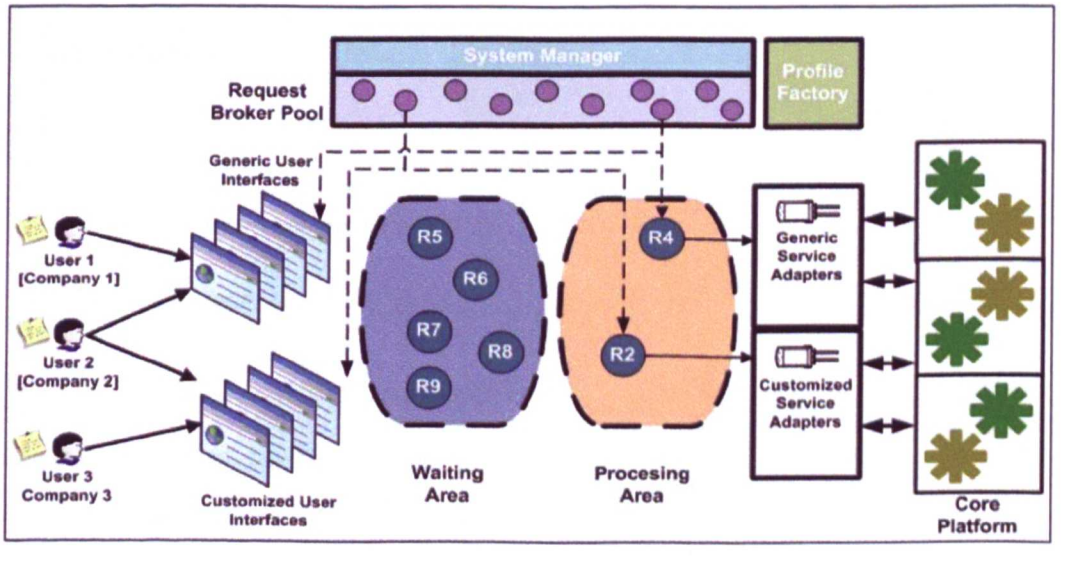


Figure 3.3: W2ASVB in workflow – Example 1

3.3.4 Request broker actions on user request

On arrival of a user request, it is queued in the waiting area as illustrated in Figure 3.3. The system manager continuously checks the waiting area and as soon as a request arrives, it is allocated a free request broker. This allocated request broker moves the request from the waiting area into the processing area and starts analysis of a request header to find out the source details, such as the name of the user interface from which the request is being generated and the action it requires. On identification of source and action, it starts searching to find out the name of the matching service adapter in the user session profile. The user session profile is a profile carried by each user and was initially allocated to each user by the system manager when they first logged in.

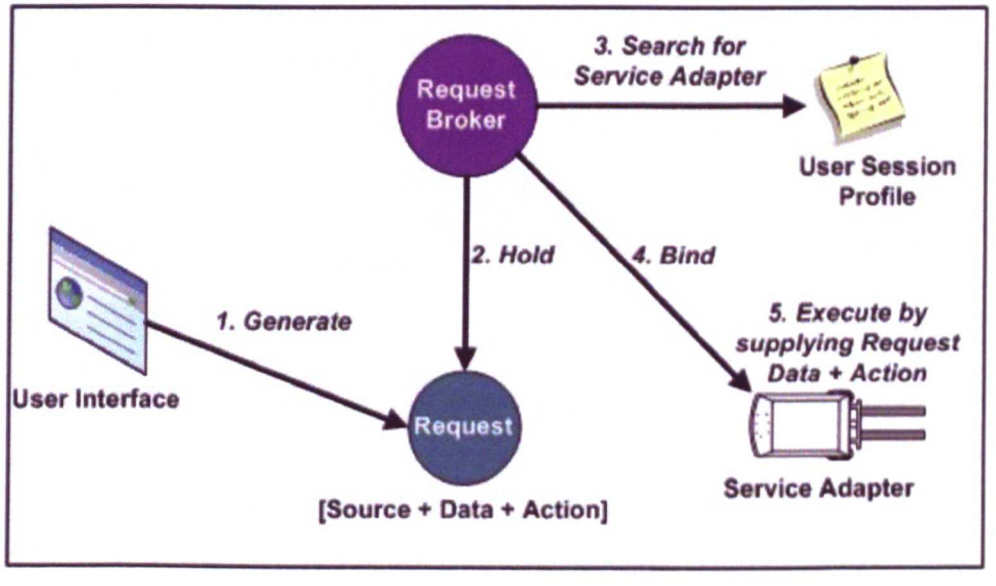


Figure 3.4: Request brokering process on request (Binding data with service adapter)

On a successful match, the request broker binds the relevant service adapter (either generic or customized) and executes it by providing the *data* and *action* parts of the request. This binding of the service adapter is called request

brokering. The service adapter only requires *data* and *action* to be executed. Hence they are services that are independent of their usage scenario and can be used by any user. This is one of the major features of the proposed W2ASVB model framework and it promotes multi-company sharing of service adapters (except customized ones because they are company-specific implementations and are only shared by the users of that particular company). Figure 3.4 illustrates the flow of control in the request brokering mechanism.

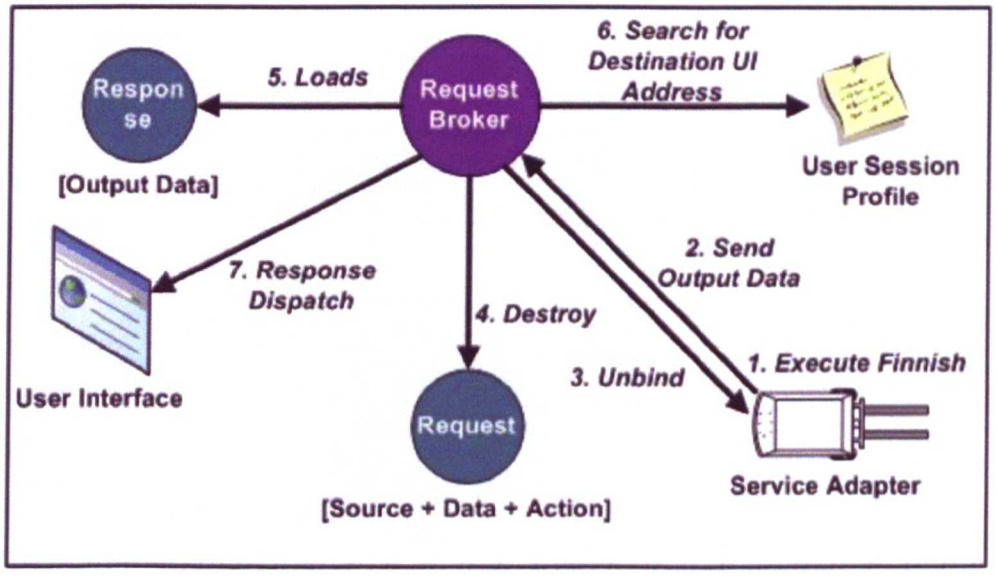


Figure 3.5: Request brokering process getting data from service adapter

On completion of the execution of the bounded service adapter, the request broker unbinds the service adapter and loads the output data (if any) which has been generated in response to the request object. It then starts searching the user session profile to find the destination user interface address. On a successful match, it dispatches the response data back to the user along with any output data. This unbinding of the service adapter and mediation of the response back to the user is called response brokering. Since the user interfaces that generate the

request and receive the response data are independent of the company's usage scenario, they can be used by any user. This is a second major feature of the proposed W2ASVB model framework as it promotes multi-company sharing of user interfaces. Figure 3.5 illustrates the flow of control in response to the brokering mechanism.

At the end of response brokering, the request broker releases all the holding resources and makes itself available to the system manager to be allocated to another request.

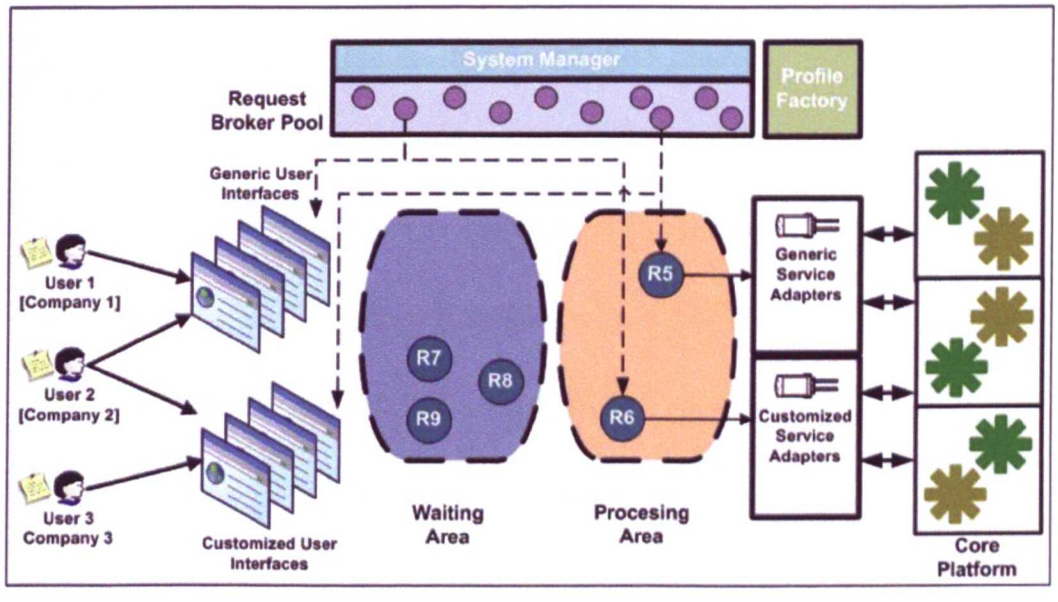


Figure 3.6: W2ASVB in workflow – Example 2

Figure 3.3 and Figure 3.6 illustrates four examples of request mappings. In both figures, user 1 of company 1 uses generic user interfaces only. Whereas user 2 of company 2 uses some generic and some customized user interfaces and finally user 3 of company 3 uses customized user interfaces only. Figure 3.3 depicts mapping of request R4 (generated by generic user interfaces) to generic service

adapter and request R2 (generated by customized user interface) to customized service adapter. On the other hand, Figure 3.6 depicts changes in the scenario where request R5 (generated by customized user interface) is mapped to the generic service adapter and request R6 (generated by generic user interface) is mapped to customize service adapter. However, the given mapping examples are not the boundaries of the model; this model is also capable of supporting other mappings such as the case of both users' requests being mapped to generic service adapters etc. the following are all possible combinations that this model can support.

All Possible Combination Supported by W2ASVB Model

Company → All Generic User Interfaces + All Generic Service Adapters

Company → All Generic User Interfaces + All Customized Service Adapters

Company → All Customized User Interfaces + All Generic Service Adapters

Company → All Customized User Interfaces + All Generic Service Adapters

Company → (Some Generic + Some Customized User Interfaces)

+

(Some Generic + Some Customized Service Adapters)

3.3.5 Profile Management Technique

Profiling techniques have proven their success in scenarios where user service requirements need to be modelled in a sophisticated manner, such as in (Howard & Kerschberg, 2004). In addition, XML is considered as a most powerful and self-descriptive language in representing complex SOA related data structures for

brokerage. It has also proven its successful applications in many real-world scenarios, such as in (Shafiq, Ding, & Fensel, 2006), (Howard & Kerschberg, 2004) and (Lee, Kuk, Kim, & Park, 2007). The same sort of technique has been applied in the proposed model with the following modifications:

- In the proposed system, the profile factory holds all the companies' profiles and each company profile only contains company information, its service adapters and user interface mapping details. It does not contain user information.
- Direct access to companies profile by request brokers is restricted.
- Direct access to companies' profiles by system manager is restricted. Instead it can only request profile factory to make changes.

Figure 3.7 illustrates the functionality of profile factory in the proposed model. Visual representation of profile factory is not presented in the proposed model as it is an integral part of management capabilities component.

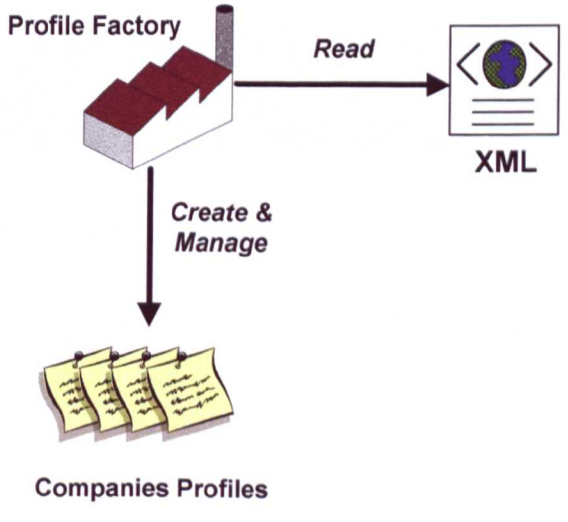


Figure 3.7: Profile factory in action

3.4 System component structure

UML is a language that defines a standard way of representing such components and their internal/external relationships with other components. There are two ways to represent the system components, namely layered representation and package representation. Layered representation is selected for explaining the overall system.

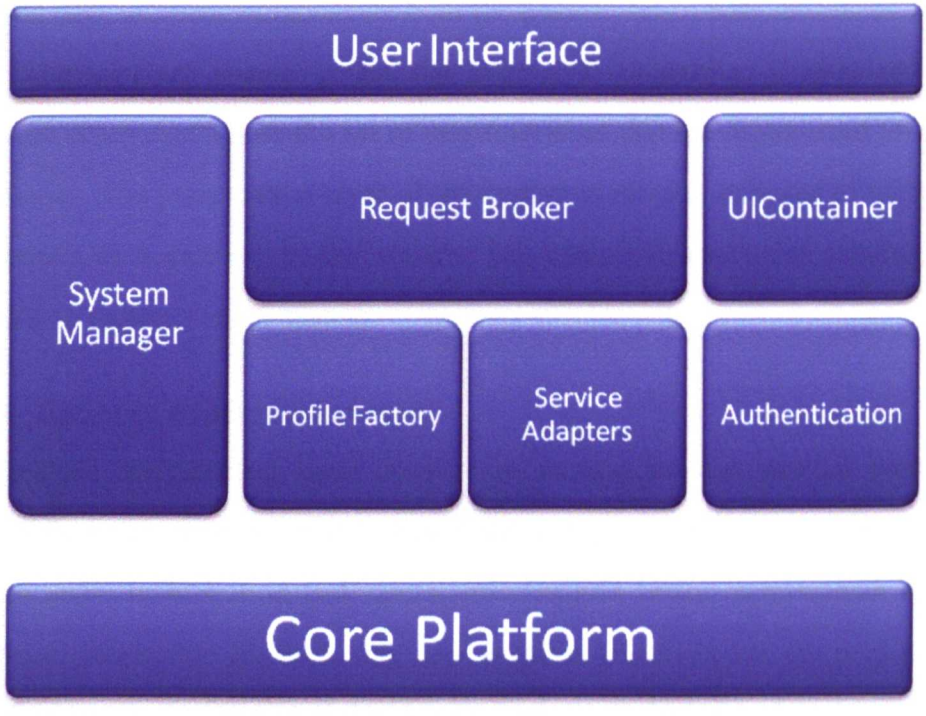


Figure 3.8: System components – Layered representation

Figure 3.8 depicts a layered representation of the system components. The overall system is divided into the following four layers:

- Presentation Layer
- Request Management Layer

- Operational Layer,
- Core Service Layer

3.4.1 Presentation layer

The presentation layer is the first layer of the model framework and it consists of a user interface component. Internally this component is divided into two sub components namely Generic user interfaces and Customized user interfaces. Generic user interface components hold the generic set of user interfaces. These user interfaces are shared across the interested companies. Customized user interface components hold the set of user interfaces that are customized for some companies. The presentation layer of the system is directly exposed to users and this serves as a gateway to the system. The users will use it for sending data processing requests to the system.

3.4.2 Request management layer

The request management layer consists of request brokers and UI container components. Presence of the system manager component on this layer indicates a control and administration task operations. Request broker components on this layer are responsible for brokering user requests (received via the presentation layer) to service adapters, and then brokering back the responses to the users via the presentation layer. The UIContainer serves as a data container that holds the response data.

3.4.3 Operational layer

The operational layer consists of a profile factory, service adapters and authentication components. The profile factory component is responsible for holding the profiles of the client companies. The authentication component is

part of a security system and provides assistance to the system manager by authenticating and authorizing users and their respective locations.

3.4.4 Core services layer

The last layer is the core services layer and it consists of core platform components. This layer provides access to core business and persistence services to the components that exist in the operational layer. Moreover, it assists the operational layer components in the accomplishment of their required functionalities. For example, it facilitates service adapters in making business services related calls for data storage and retrieval, and it also assists the authentication component in validating user credentials such as user id, password and roles. In addition, it also provides services to the profile factory component for retrieval of company-related information from the database such as location details, employee details, addresses, contacts numbers etc.

3.5 System use-case model

During the detailed analysis phase of W2ASVB all the possible use-case models for the framework were considered. The primary focus during the design, implementation and validation phases will be on the proof of model framework.

3.5.1 General request use-case model

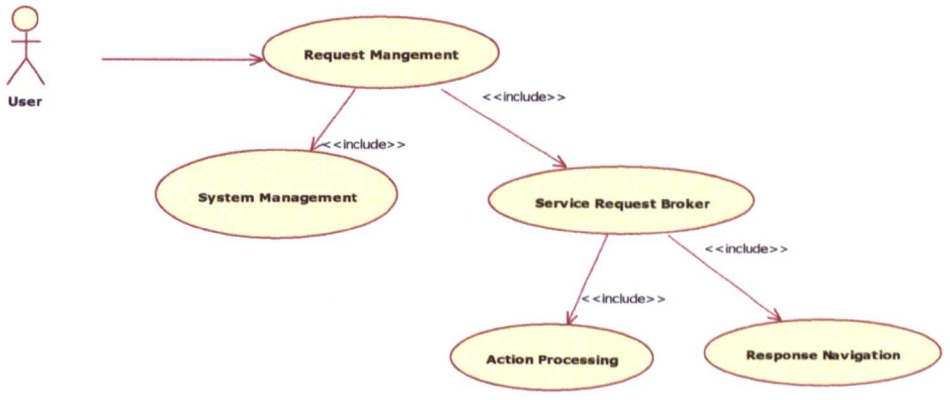


Figure 3.9: Request use-case model

3.5.2 General request use-case descriptions

The use case displayed in Figure 3.9 represents the general flow of all requests in the proposed model framework. All the users' requests such as process sales orders, query about the stock quantity etc are shown by this use case, but on the basis of the logged in user profile, "Action Processor" performs the different operations with the help of system core functionality. Figure 3.10 displays the activity diagram of the above use case diagram. It shows the functionality of the internal request process.

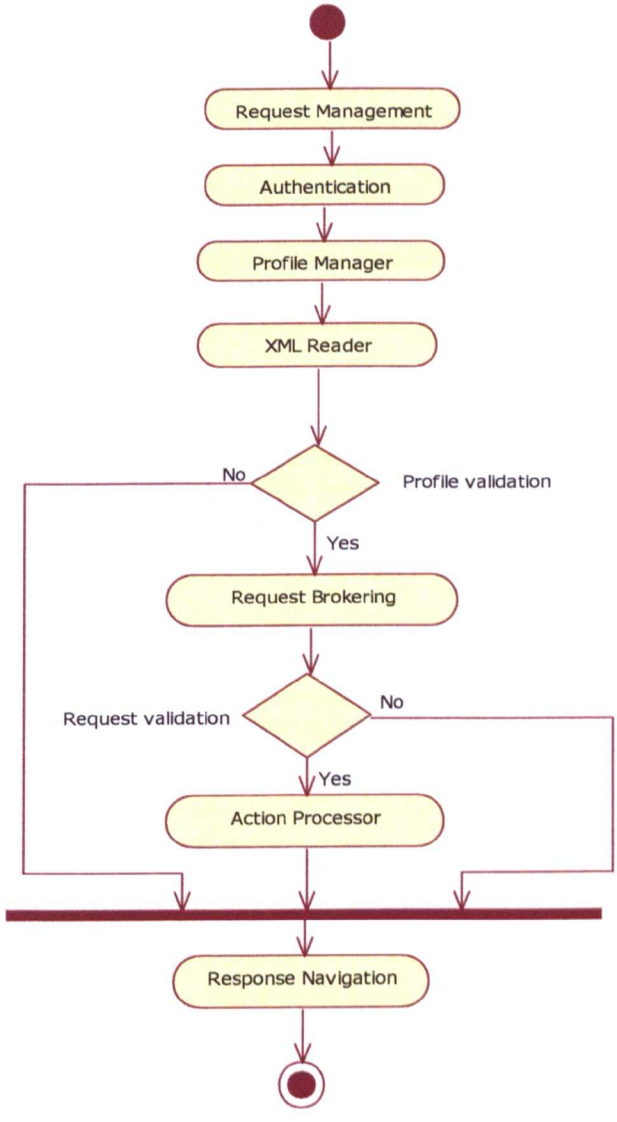


Figure 3.10 Request activity diagram

Use case: Request Management
ID: UUC1
Brief Description: Registered user enters his/her requests and gets the response on the basis of his/her profile and request with the help of the system.
Primary Actors: User (Registered user of the system).
Secondary Actors:

None.
Preconditions: Registered user should be logged in into the system.
Main Flow: 1. The use case starts when the user enters the request into the system (Request can be 'Make sale', 'Process order' etc). 2. The system transfers the user request to "Request management" area. Requests are queued in this area for processing. 3. The "Request management" area transfers the user information to "System Management". 4. "System Management" allocates the request to the "Service Request Broker". 5. "Service Request Broker" processes the request on the basis of the "System Management" output and allocates the request data with desire action instructions to the "Action Processing". 6. "Action Processing" transfers the user's request to the "Response Navigation". "Response Navigation" returns the final response to the user.
Postconditions: The user gets the response of his request.
Alternative Flow: 1. "Service request broker" objects are not available. 2. User does not have privilege for this action. 3. Invalid values in request.

Alternative flow: Request Management: "Service request broker" objects are not available
ID: UUC1.1
Brief Description: The system informs the user that all resources are busy and request will process in few minutes. (It is important to inform the user and keep him/her update in order to keep the connection active in web browser).
Primary Actors: User
Secondary Actors: None.
Preconditions: Registered user has entered a request to the system.
Alternative Flow: 1. The alternative flow begins after step 4 of the main flow. 2. The system informs the user that his/her request is in a queue and for that moment no resource is available for processing.
Postconditions: 1. The system updates the user about the request progress. 2. The user will have an option to cancel the request and process it later.

Alternative flow: Request Management: User does not have privilege for this action
ID: UUC1.2
Brief Description: The system informs the user that his/her profile is not able to perform this request.
Primary Actors: User
Secondary Actors: None.
Preconditions: Registered user has entered a request which is not allowed in his/her profile (like sales order of more than £10,000 can only be processed by managers level profile, while, staff level profile users are not allowed to process the sales orders of this worth).
Alternative Flow: 1. The alternative flow begins after step 3 of the main flow. 2. The system informs the user that his/her profile has no privilege to perform the request.
Postconditions: 1. System does not processe the user's request. 2. The system asks the user to re-enter the request.

Alternative flow: Request Management: Invalid values in request
ID: UUC1.3
Brief Description: The system informs the user that his/her request has invalid values.
Primary Actors: User
Secondary Actors: None.
Preconditions: The user has attempted to process the request without providing the require information.
Alternative Flow: 1. The alternative flow begins after step 5 of the main flow. 2. The system informs the user that his/her request has missing information.
Postconditions: 1. The system asks the user to re-enter the value(s) in the request.

3.5.3 Sales person use case diagram

Following are few use case diagrams for sale person on the system built by using the W2ASVB model framework.

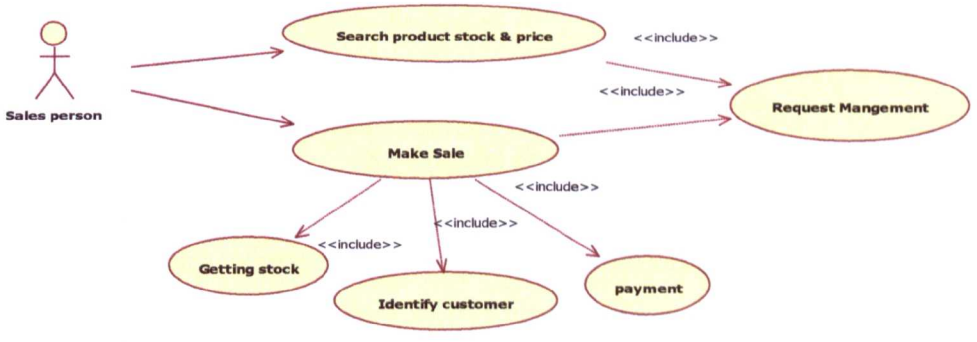


Figure 3.11: Sales person use-case model

Use case: Search Product Stock & Price
ID: SPUC1
Brief Description: Sales person checks a product price and availability in stock.
Primary Actors: Sales person
Secondary Actors: None.
Preconditions: Sales person is logged in into the system and have privilege to access the product.
Main Flow: 1. The use case starts when the sales person selects "Search Product Stock & Price". 2. Sales person enters the criteria to search for product details. 3. System passes the user's request to the "Request Management". "Request Management" passes the request to the "System Manager" for request permission on the basis of the profile. 4. "System Manager" transfers the user's request to "Service Request Broker" after getting the valid permission. 5. "Service Request Broker" allocates the service adapter to the request on the basis of the request and profile of the logged in user. If the profile is configured to use a custom search for products then it allocates the customized service adaptor for process otherwise generic service adaptor. All these service adaptors exist into the "Action Processing". 6. After getting a response from the core functionality, appropriate

service adaptor will transfer the response to the "Response Navigation". 7. The system displays a list of locations containing the required products and respective stock with the price for each location.
Postconditions: 1. The sales person gets information about the current stock situation at each location with price.
Alternative Flow: No Product Found (Invalid product). Out of stock.

Alternative flow : Search Product Stock & Price: No Product Found
ID: SPUC1.1
Brief Description: The system informs the sales person that the product does not exist in the system.
Primary Actors: Sales person
Secondary Actors: None.
Preconditions: 1. Sales person is logged in into the system and has a privilege to access the products. 2. Sales person searches product by providing required criteria.
Main Flow: 1. The alternative flow begins after step 6 of the main flow. 2. System informs sales person that the requested product does not exist in the system.
Postconditions: The sales person is informed that the requested product is out of stock and is subsequently asked to re-enter the search criteria.
Alternative Flow: None.

Alternative flow : Search Product Stock & Price: Out of stock
ID: SPUC1.2
Brief Description: The system informs sales person that the product is out of stock.
Primary Actors: Sales person
Secondary Actors: None.
Preconditions: 3. Sales person is logged in into the system and have a privilege to

access the products. 4. Sales person has searches the product by providing required criteria.
Main Flow: 3. The alternative flow begins after step 6 of the main flow. 4. System informs the sales person that the requested product is out of stock.
Postconditions: The sales person is informed that the requested product is out of stock and is asked to re-enter the search criteria.
Alternative Flow: None.

Use case: Make Sale
ID: SPUC2
Brief Description: Sales person wants to record a sale in the system.
Primary Actors: Sales person
Secondary Actors: None.
Preconditions: Sales person is logged in into the system and have privilege to log sales activity. Customer does the payment in cash.
Main Flow: 1. The use case starts when the sales person selects 'Make Sale' option. 2. Sales person records all the sales item(s) one by one by using a scanner or manual data entry (providing item number). 3. Sales person enters the customer information. 4. Sales person enters the amount; customer has paid and presses the process button. 5. System sends the request to the "Request Management". 6. "Request Management" sends the request to "System Manager" and then "System Manager" allocates the request to "Service Request Broker". 7. After getting a response from the system, "Service Request Broker" transfers the response back to the user. 8. If everything is fine in the order then the system displays the order information with the remaining customer's price. 9. The system gives an option to prints out the current sales receipt.
Postconditions: 1. The new sale has been successfully recorded in the system. 2. The quantity in stock of the relevant product is updated.
Alternative Flow: Cancel Sale.

Alternative flow: Make Sale : Cancel Sale
ID: SPUC2.1
Brief Description: The sales person cancels the 'Make Sale' process.
Primary Actors: Sales person
Secondary Actors: None.
Preconditions: The sales person is logged in into the system and has privilege to log the sales activity. Customer does the payment in cash.
Alternative Flow: 1. The alternative flow begins at any time before step 4. 2. The sales person cancels the 'Make Sale' process.
Postconditions: 1. The 'Make Sale' process has not been recorded in system. 2. System asks the sale person to re-enter the sale information or navigate to the main screen.

3.6 Conclusion

This framework will serve as a mediation platform for request brokers. It will also provide a high level of abstraction by encapsulating low level details of the system such as request handling, request mediation, response handling, service loading etc. Implementation of this proposed model framework will result in a prototype version of the system. This version will be verifiable by the user and its designed parameters can also be validated by the service providing firm.

4. Design of a W2ASVB Model Framework

4.1 Introduction

The concept of a new proposed model along with a very high level description of its internal functioning has been discussed. From this point onwards, a system is required to be successfully built upon the proposed model framework to validate the model framework objectives. It is worth mentioning here that although the model framework itself does not contain any use-cases, the internal functionalities of the model framework will be triggered on the initiation of use-cases of the system. To this end, subsequent sections present detailed functional descriptions of the proposed model framework.

4.2 An enhanced web 2.0 e-business framework

W2ASVB with fundamental changes provides an effective model framework which serves as a front and middleware collaboration model between the core platform and client operational requests. This model framework will provide an intelligent way of providing interfaces for sharing core business logic among small businesses and at the same time providing customization facility of their individual core business logic without affecting other businesses. The proposed model framework exhibits such capabilities and is intelligent enough to handle such complex scenarios. Therefore, this new model framework needs to be implemented on an initial version to evaluate its working capabilities.

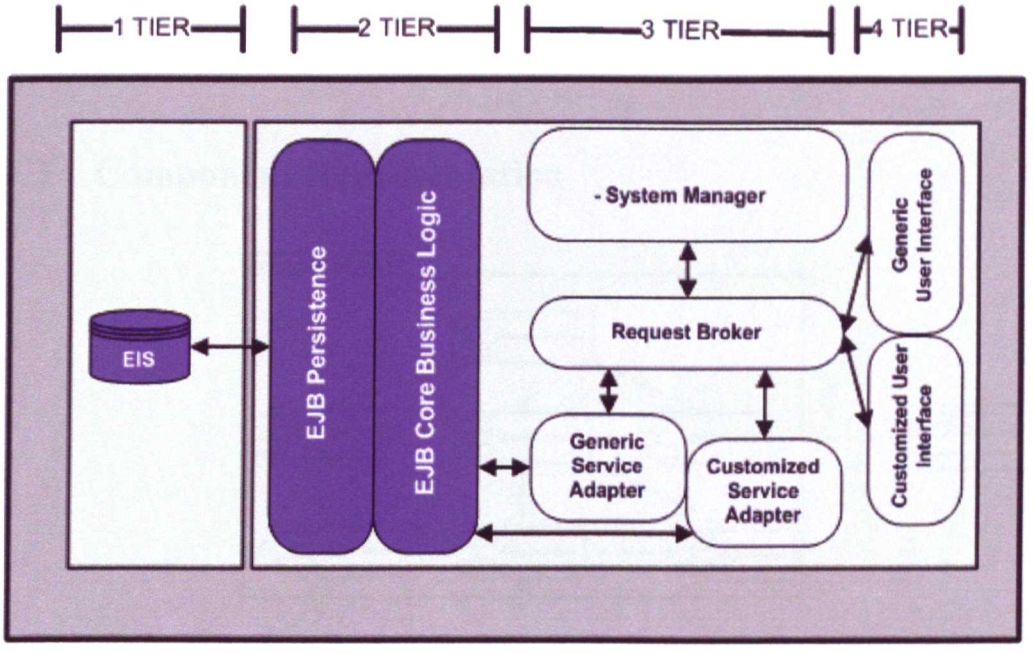


Figure 4.1: An architectural design of a modified Web2.0 base model framework

The output of the application of the proposed model framework on a multi-tier Web2.0 based framework is presented in Figure 4.1. The fundamental changes of the model framework are being integrated in between tier 3 and 4 and serve as a front and middleware collaboration model. The framework makes use of request brokers to intelligently route the users' request to appropriate service adapters. On the completion of execution of that service adapter, it intelligently routes the response back to the appropriate user interface. Figure 4.1 illustrates the working relationship between different components of the system. Each component in the system is developed as a standalone entity and is capable of performing its functionality without being tightly coupled with other components. This loosely coupled nature of the system components eases the development and maintenance activities. The detailed discussion about each

component including their internal structures and functionalities are presented in the sections below.

4.3 Component Representation

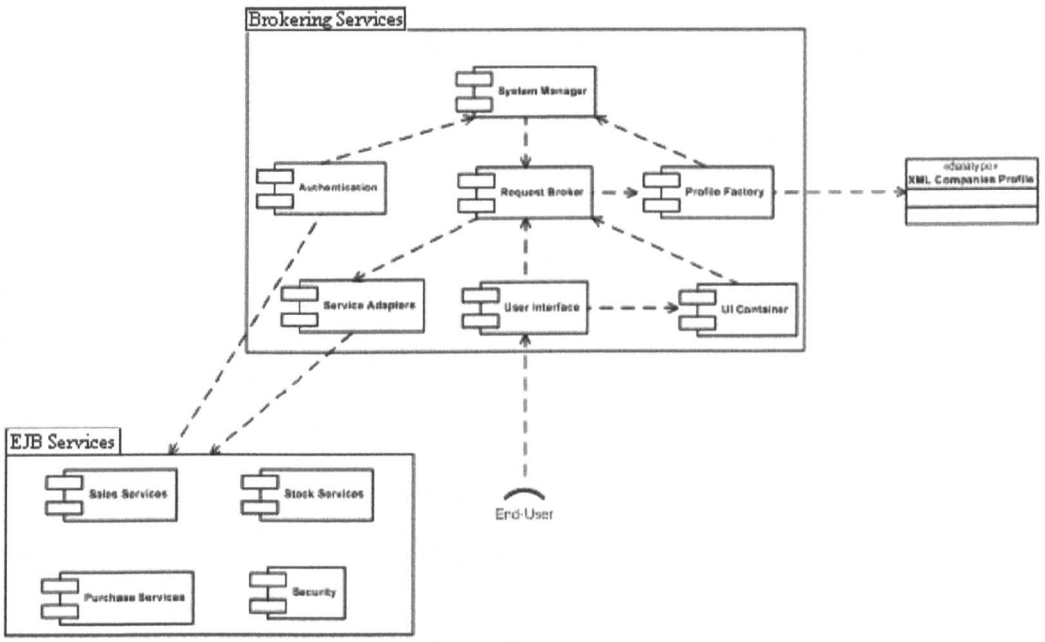


Figure 4.2: System components

Figure 4.2 illustrates a component representation of the overall system. The above group of components is named as Brokering Services. Details with regards to the relationship between components such as internal interfaces with other components and external interfaces with the outside world are also presented in the Figure. A detailed discussion about each component, including its internal class structure and functionalities, is presented in the following sections.

4.3.1 System manager component

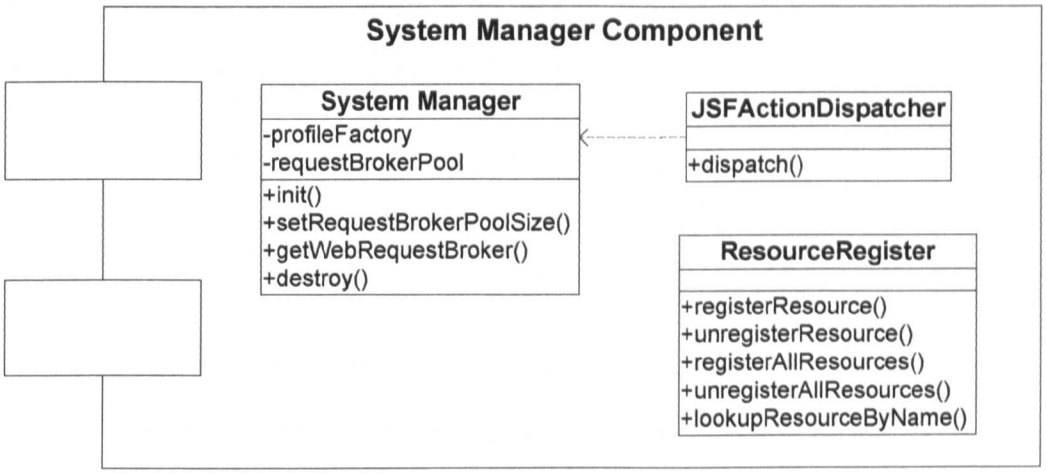


Figure 4.2: System Manager Component sub-modules

This component is a controller component of the system. It consists of three classes, namely SystemManager, ActionDispatcher and ResourceRegister. The functionalities of these classes are given below.

The JSFActionDispatcher class is a gateway module of the system. Its *dispatch()* method contains the business logic to obtain the free request broker from the system manager, and then dispatches the request processing responsibility to the allocated request broker. This method is called by the user interface when a user sends a data processing request to the system.

The ResourceRegister class holds the references of all the services of the core platform in a "Name-to-Service"-like mapping mechanism. Any object that needs to execute the core platform services provides the name of the required service-id to the ResourceRegister, and in response is given the reference of the requested service. When the object finishes using the service, it returns the service reference to the ResourceRegister.

The SystemManager class is responsible for controlling the overall system. This module loads immediately at the deployment of the system on an application server. It is mainly responsible for the following activities:

- Initialization of a request broker pool and its population with the initialization of individual request brokers. The business logic for this activity in *init()* method.
- Initialization of the ResourceRegister; the business logic for this activity is initialized at the loading of the system Manager module.
- Initialization of a profile factory; the business logic for this activity is given in *init()* method.
- Destruction of the ResourceRegister, request brokers pool and profile factory when system is terminated; the business logic for this activity is given in *destroy()* method.
- Allocation of a free request broker to the JSFActionDispatcher; on request from the JSFActionDispatcher, the SystemManager searches the pool for a free request broker and allocates one for a request. The *getWebRequestBroker()* method of the SystemManager class contains the business logic to perform this function.
- Manually increasing or decreasing the size of the request broker pool on request from the system administrator; the *setRequestBrokerPoolSize()* method contains business logic to perform this function.

To elaborate further on the above outlined interactions, UML sequence diagrams are presented in section 4.6.1.

4.3.2 Profile Factory Component

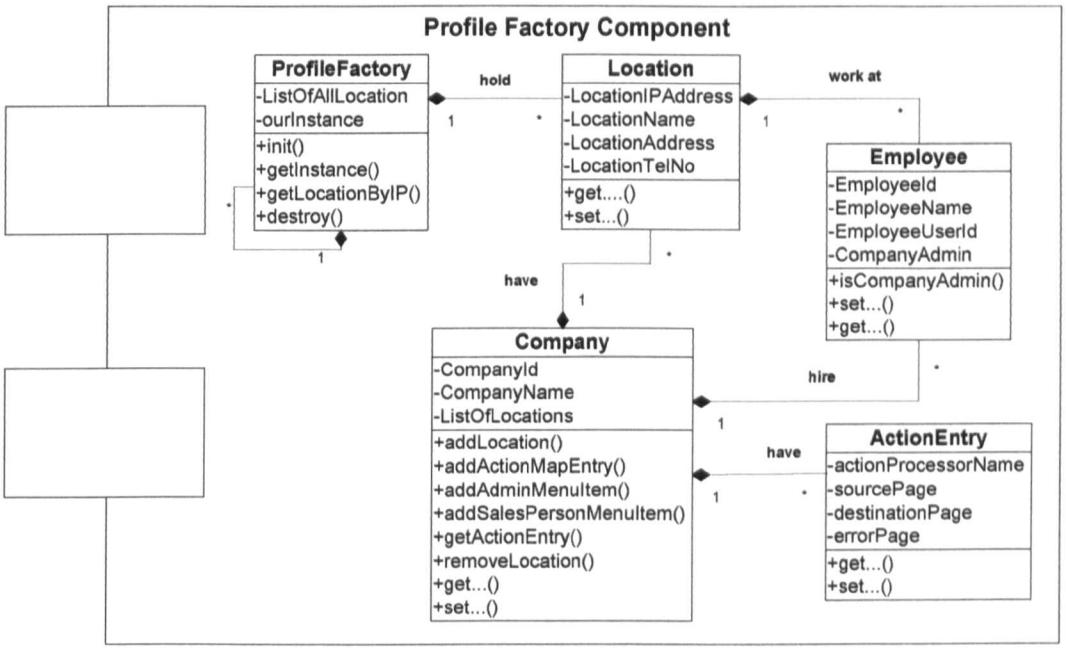


Figure 4.3: Profile Factory sub-components and their relationship

The Profile Factory is one of the most important components of the proposed model framework; it holds the data dictionary which represents the meta-data about the client's companies. This component consists of five core classes, namely ProfileFactory, Company, Location, Employee and ActionEntry. The last four classes collectively represent the companies' profiles. The ProfileFactory class acts as a manager, and is responsible for the creation, allocation and management of these profiles.

As mentioned earlier, a client of the system is a retail company running its business on several locations. A location represents a retail shop (i.e. point of sale) where a company sells its products. The company class of this component represents a retail company, and holds the company-related data, such as a list of its locations. The location class of this component represents these locations and

holds the location-related data. The relationship between a company and a location class is one-to-many. The Employee class represents a person who works for a company. The relationship between a company and employee class is one-to-many. Note that the owner of a company is also modelled as an Employee of the company; this is because the system will be used by small retail companies and the owners of such companies also need to work in the shops. The *CompanyAdmin* attribute of the Employee class is a Boolean flag that distinguishes between ordinary employees and the company owner. In addition to performing ordinary retail functions, the owners of a company have additional responsibilities such as administration and management of the company, its employees and information on its location; for this reason they will have a separate set of user interfaces to perform these functions.

The ActionEntry class is a key class in this component. A single instance of this class holds the mapping instruction of one business activity for any single company. In other words, it contains a detailed mapping from one source page to a service adapter and then from the service adapter to a destination or error page. Service adapters in the system are modelled as ActionProcessors. The term "action processor" is nothing but an analogy of a service adapter. Since any company may have more than one business activity to the run the e-business, the Company class holds the array of ActionEntry objects. Request brokers (of the Request Broker Component) use this array of action entry objects for dynamic request and response routing.

All locations along with other related details are embedded in the meta-data file as shown in Figure 4.4. This file is then supplied to the ProfileFactory class for parsing. The ProfileFactory parses the information from the XML file and creates

the corresponding Company and Location objects. This parsing result in the form of a Key-Value type list, where Key represents a location id and Value holds the reference of an object in memory.

```
<?xml version='1.0' encoding='utf-8'?>
<Company-Profiles>
  <Company id="61" url="www.ca-electronics.com">
    <Locations>
      <Location IP-Address="217.35.95.208" logopath="resources/logodir/CompanyName1-LocationName1.gif"/>
      <Location IP-Address="217.45.161.252" logopath="resources/logodir/CompanyName1-LocationName2.gif"/>
      <Location IP-Address="217.35.95.138" logopath="resources/logodir/CompanyName1-LocationName2.gif"/>
    </Locations>
    <Action-Processors-Mapping>
      <Action-Entry source="genericMakeSale.jsp" dest="showGenericPrintInvoice" actionProcessor="generic.InvoiceProcessor" errorpage="showGenericMakeSaleErro
      <Action-Entry source="genericStockCheck.jsp" dest="NULL" actionProcessor="generic.stock.StockCheckProcessor" errorpage="NULL" />
      <Action-Entry source="genericSupplierRelation.jsp" dest="NULL" actionProcessor="generic.supplier.SupplierProcessor" errorpage="showSupplierRelationErro
      <Action-Entry source="GenericAddProduct.jsp" dest="messages" actionProcessor="ProductProcessor" errorpage="error" />
      <Action-Entry source="GenericAddCustomer.jsp" dest="messages" actionProcessor="CustomerProcessor" errorpage="error" />
      <Action-Entry source="GenericAddSupplier.jsp" dest="messages" actionProcessor="SupplierProcessor" errorpage="error" />
    </Action-Processors-Mapping>
    <Admin-Menu-Mapping>
      <Menu name="desktop" navigation-case="showGenericAdminDesktop"/>
      <Menu name="recorddelivery" navigation-case="/samand/faces/genericRecordDelivery.jsp"/>
      <Menu name="reprintlabels" navigation-case="/samand/faces/genericReprintLabels.jsp"/>
      <Menu name="printsalesreports" navigation-case="/samand/faces/genericPrintSalesReports.jsp"/>
      <Menu name="printpurchasereports" navigation-case="/samand/faces/genericPrintPurchaseReports.jsp"/>
      <Menu name="printstockreports" navigation-case="/samand/faces/genericPrintStockReports.jsp"/>
      <Menu name="printattendancereports" navigation-case="/samand/faces/genericPrintAttendanceReport.jsp"/>
      <Menu name="addsupplier" navigation-case="/samand/faces/genericSupplierRelation.jsp"/>
      <Menu name="updatesupplier" navigation-case="/samand/faces/genericSupplierRelation.jsp"/>
      <Menu name="closetill" navigation-case="/samand/faces/genericCloseTill.jsp"/>
      <Menu name="resetstaffpassword" navigation-case="/samand/faces/genericResetStaffPassword.jsp"/>
    </Admin-Menu-Mapping>
    <SalesPerson-Menu-Mapping>
      <Menu name="desktop" navigation-case="showGenericSalesPersonDesktop"/>
      <Menu name="stockcheck" navigation-case="/samand/faces/genericStockCheck.jsp"/>
      <Menu name="makesale" navigation-case="/samand/faces/genericMakeSale.jsp"/>
      <Menu name="managesupplier" navigation-case="/samand/faces/genericSupplierRelation.jsp"/>
      <Menu name="viewperformance" navigation-case="/samand/faces/genericPerformance.jsp"/>
      <Menu name="movestock" navigation-case="/samand/faces/genericMoveStock.jsp"/>
      <Menu name="supplierrelations" navigation-case="/samand/faces/genericSupplierRelations.jsp"/>
    </SalesPerson-Menu-Mapping>
  </Company>
```

Figure 4.4: XML file representing meta-data of a company and its associated locations

4.3.3 Request broker component

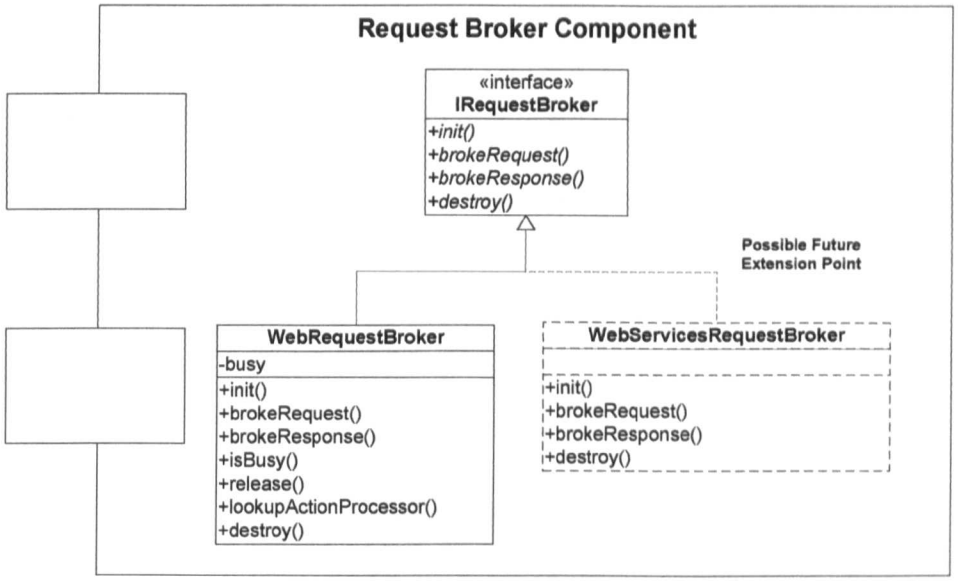


Figure 4.4: Request broker sub-modules

The Request broker component is a main operational component of the model framework. It consists of an IRequestBroker interface and the WebRequestBroker class. The WebServicesRequestBroker class is only an illustration of a possible future extension. Figure 4.4 illustrates the sub sections of the request broker.

The IRequestBroker interface has four main responsibilities, namely *init()*, *brokerRequest()* , *brokerResponse()* and *destroy()*. The WebRequestBroker class implements the IRequestBroker interface. The *init()* and *destroy()* methods of the WebRequestBroker class are lifecycle methods. These WebRequestBroker methods are active throughout the workflow, whereas *brokerRequest()* and *brokerResponse()* methods are active according to the request made. The *brokerRequest()* method contains the business logic to route the user request to the appropriate action processor, whereas the *brokerResponse()* method contains

the business logic to route the response back from the action processor to the destination user interface (destination or error web page).

4.3.4 User interface container component

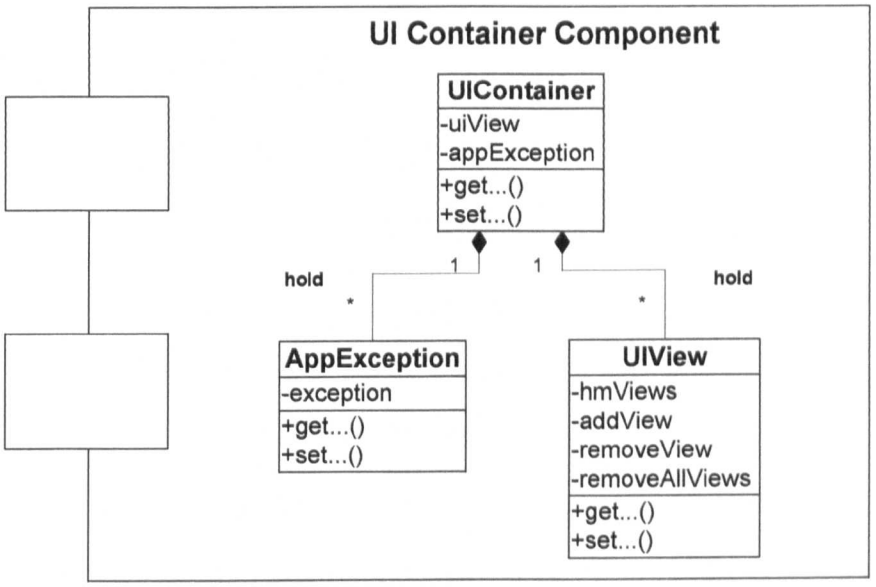


Figure 4.5: User Interface sub-modules

The new proposed model framework implements the user interface in a way that they can work in a protocol-independent manner i.e. accepting plain data objects for processing, and generating plain data objects as output. Now, the question is how this protocol-independent output data will be sent back to protocol-dependent user interfaces (i.e. web pages) to be presented to a user. The answer to this question is the deployment of the User Interface Container component. This component consists of three classes, namely UIContainer, UIView and AppException. Figure 4.5 illustrates the internal design of this component.

The Action Processor generates output data during execution; this output data is in the form of collection of data objects stored inside the UIView. Any exception (if generated) loads into the AppException object. The UIContainer

class serves as a container for both UIView and AppException. On completion of the request processing cycle, the request broker sends this UIContainer to the user interface, which then parses the stored data (of UIContainer) into a format that is presentable to a user.

4.3.5 Authentication component

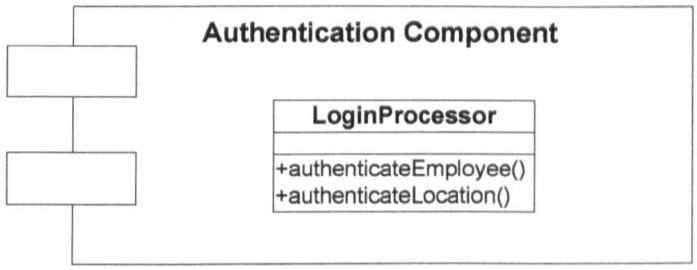


Figure 4.6: Authentication Component

The Authentication Component is a sub-component of the main security component. This component consists of only one class called LoginProcessor. There are two main responsibilities of this class; one to authenticate the employee credentials such as User Id, password and Role, and second to authenticate its location of access, at the time of login. The "authenticateEmployee" and "authenticateLocation" processes are designed to perform these functionalities. These processes connect to the core platform to verify required information. On successful authentication, the Login module (of the user interface component) creates a user session profile and loads it with the necessary object references such as employee, location, company, action processor mapping and user interface menu mapping tables.

4.3.6 Service adapter

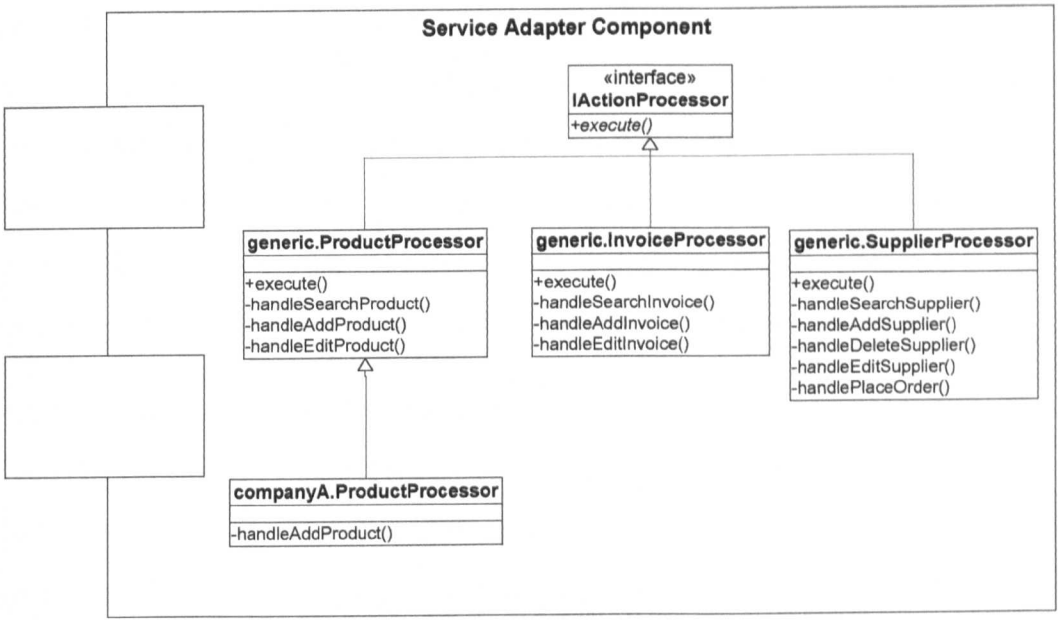


Figure 4.7: Service Adapter sub-components

The Service adapter component holds the overall workflows of the system in the form of action processors. This component consists of an IActionProcessor sub module. The IActionProcessor module is a gateway of service adapter component and also serves as a plug-in point for new action processors. It defines only one public interface process called "execute". All generic action processors use this process by providing their own generic implementation of this process. Afterwards, the name of these generic action processors is placed under the interested company (or companies) profiles within a meta-data mechanism.

Figure 4.7 illustrates the design of the three generic and one customized action processors. In addition to a standard "execute" process; these action processors implement their own private request "handle" processes. These are the special processes and are there to handle special functionalities based on user-provided action commands. For example, in the case of the Sale activity, a user submits

the *Invoice Data* and provides the *Action Command* = *ADD* in the request to the system. On arrival of the request, the request broker loads the corresponding generic invoice processor (assuming that a company is using a generic invoice processor) and runs the "execute" process command. The Execute process internally checks the given *Action Command* and switches the control to an appropriate handler process, which in this case is the "handleAddInvoice" process. This is just a design approach to separate different functionalities.

Customized action processors need not implement the IActionProcessor module directly; instead, they extend the existing relevant generic action processor that a company wishes to customize. Because of this inheritance, all the generic processes of selected generic action processor will become available for customization. For example, if a company only wants to customize the way their products are adding to the database etc; all the service provider needs to do, is to extend existing generic product processor, and overrides its "handleAddProduct" mechanism. Subsequently the name of old generic action processor needs to be replaced by a new action processor name in the meta-data module. The rest is automatically handled by the system. In Figure 4.7, the Company A's product processor is an illustration of such customization.

To elaborate further on the above outlined interactions in the sample example, UML sequence diagrams are presented in section 4.6.3.

4.3.7 User Interface Component Design

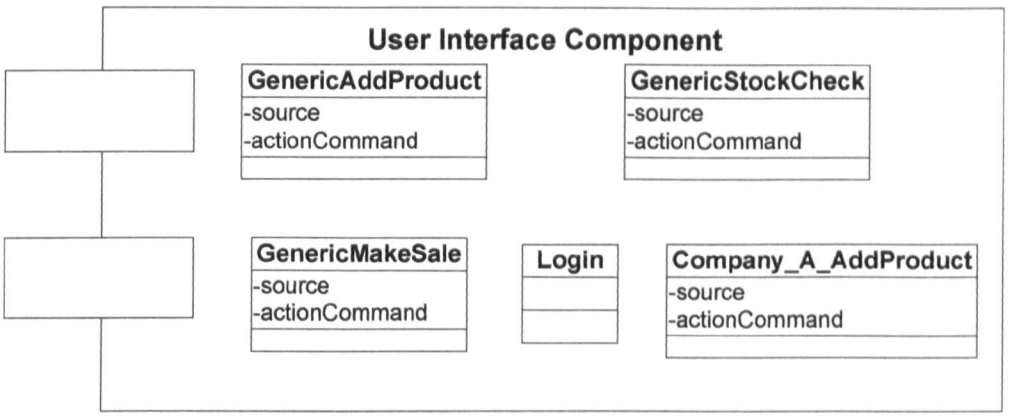


Figure 4.8: User interface component designnent design

The User interface component of the system contains user interfaces (i.e. web pages or views). Both the generic and customized user interfaces are set up together in this single component, as illustrated in Figure 4.8. These user interfaces can have any number of attributes with any number of methods. The only exception is that, they must contain *source* and *actionCommand* attributes. The *Source* attribute indicates the name of the page that submits the data to the system, whereas the *actionCommand* attribute indicates the user action required on submitted data.

4.3.8 Systems Class Structure

Figure 4.9 illustrates the structure of the classes in the proposed model framework modules.

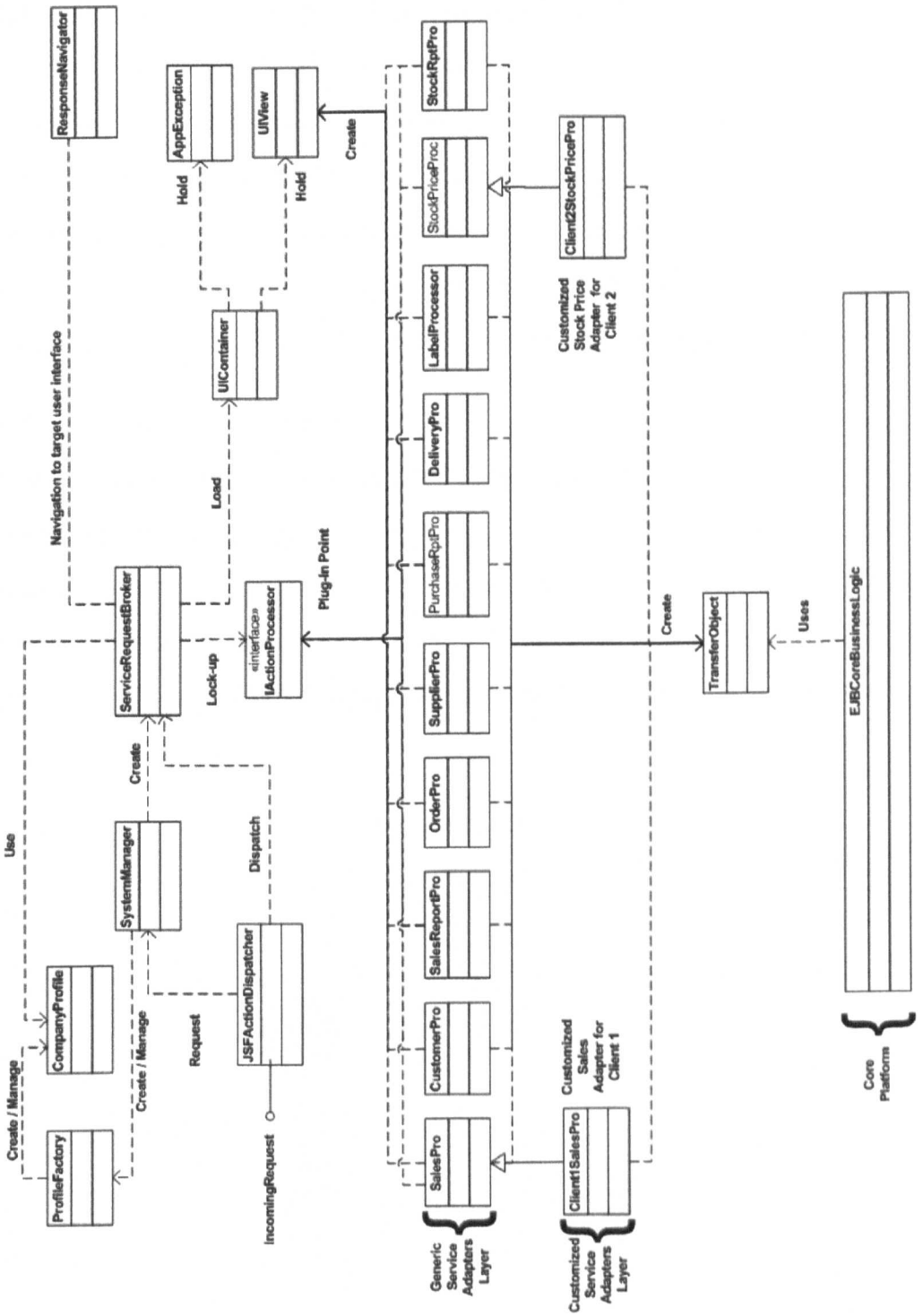


Figure 4.9: Systems class structure

4.4 Request routing technique explanation

A better understanding of the overall component structure of the model framework has been developed in the above sections. However, no discussion has been provided on how the different modules of these sub-modules interact with each other to perform certain functionalities. To understand the core functionality of the system, one needs to understand the request routing technique. The following sections provide further details on request routing processing.

4.4.1 Action management

After successful authentication of a user, the login class creates the user session profile and allocates it to the user. This user session profile (along with other details) holds the action processing mapping mechanism. Figure 4.10 illustrates action processor mapping mechanism of a user session profile. This action processor mapping mechanism helps the system decide about a user's request and response routing paths.

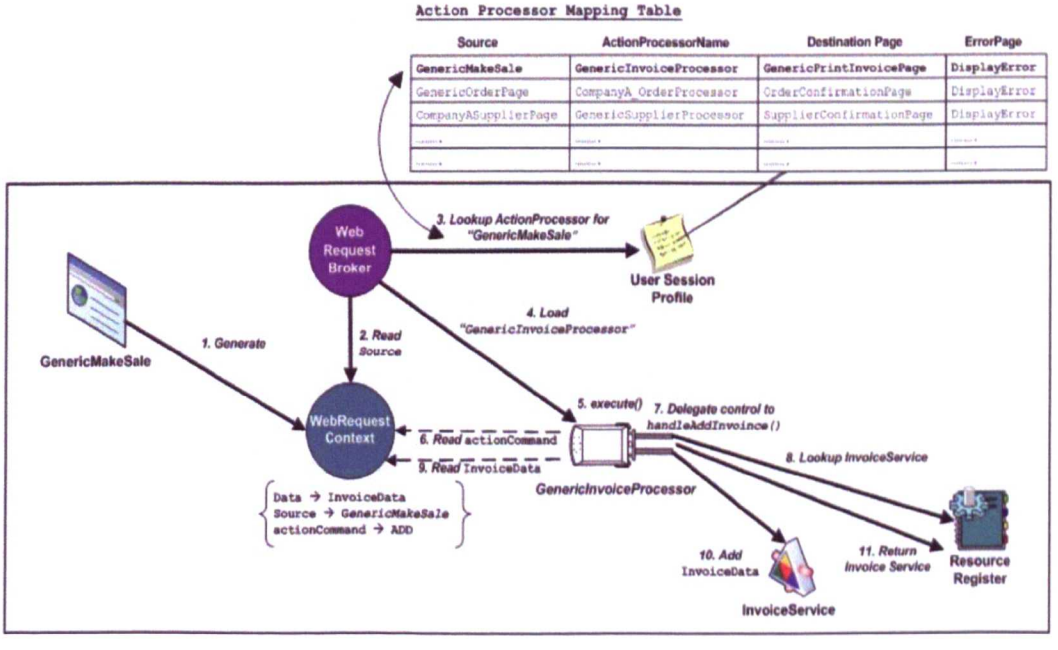


Figure 4.10: Action management activity for make sale process

Figure 4.10 depicts a sale process. This initiative was taken due to the irreducible complexity of the scenario and to provide better understanding to the reader. In the sale activity, by using the GenericMakeSale user component, a sales person (i.e. a user) submits the customer invoice data to the system. On arrival of this user request, the system creates the WebRequestContext object and fills it with user-submitted information. This WebRequestContext object encapsulates three important pieces of information, namely data, source and actionCommand. The description of each part is as follows:

- The data part represents the InvoiceData that was keyed in by the user.
- The source part represents the name of the page from which the request is being generated; and it is GenericMakeSale in the current scenario.
- Finally, the actionCommand part specifies the action that the user wants the system to take on the submitted data. It is ADD in this current scenario; which means that the user wants to save the InvoiceData in the database.

Subsequently, the system manager allocates this user request to a free web request broker. This web request broker uses the WebRequestContext object for mapping the user request to an appropriate action processor. Figure 4.10 explains the complete scenario. This phase is called Action Management. The next phase is view management; it describes how the output data that is being generated in response to the execution of the action processor is routed back to the user.

4.4.2 View management

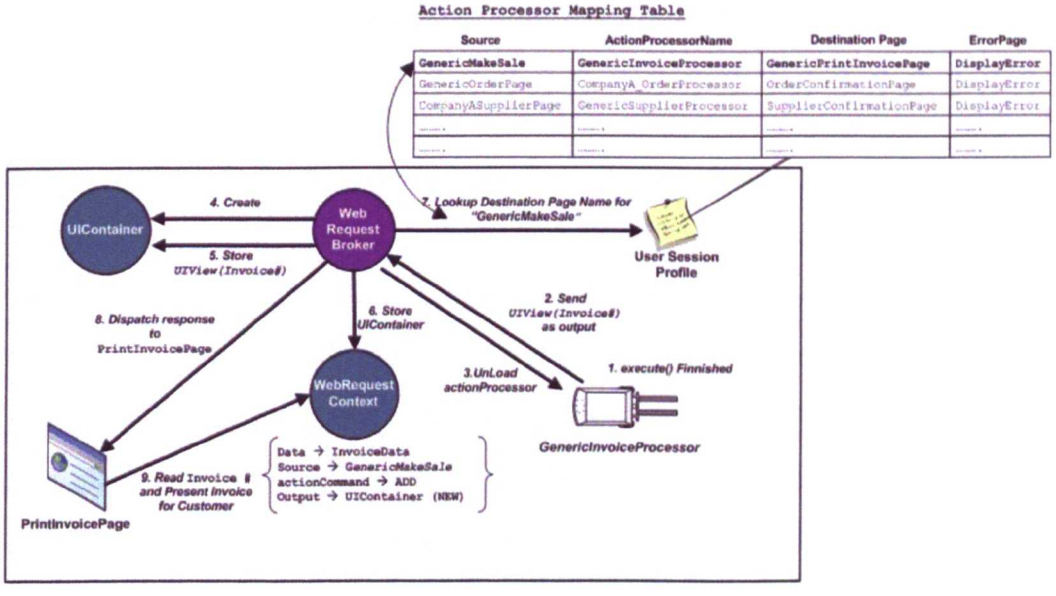


Figure 4.11: View management activity for make sale process

Figure 4.11 illustrates the view management phase of the sale process. The execute process of the GenericInvoiceProcessor generates the output data in the form of a UIView. The Web request broker creates the UIContainer and stores this UIView within it (in the current case, the UIView holds the Invoice number of the newly saved invoice). Eventually the UIContainer instance is stored inside the web request context object.

The web request broker then searches for the corresponding destination page name in the action processor mapping table. Once the destination page object is found, the request broker dispatches the response to the destination page container which is called PrintInvoicePage in the current scenario. Note that, the names of the destination and error pages in the action processor mapping module are abstract names e.g. GenericPrintInvoicePage. Conversion of these abstract names into absolute paths is the responsibility of the Navigator module. Finally,

PrintInvoicePage reads the output data (i.e. Invoice number) from the web request context object and presents a printable sales' invoice to the user. Figure 4.11 explains the complete scenario.

4.5 Business logic on-demand customization mechanism

This is a key question based on which the investigation started. The answer to this question is hidden in a meta-data module and its processing. As described earlier, the CompanyProfile module and its mapping mechanism are the data dictionary of this system and a profile factory which is a controller that uses this data dictionary to provide direction to others. Subsequent sub-sections presents examples of both business logic and user interface customizations.

4.5.1 Customization of business logic

Company A : Action Processor Mapping Table (BEFORE)

Source	ActionProcessorName	Destination Page	ErrorPage
GenericMakeSale	GenericInvoiceProcessor	GenericPrintInvoicePage	DisplayError
GenericOrderPage	GenericOrderProcessor	OrderConfirmationPage	DisplayError
CompanyASupplierPage	GenericSupplierProcessor	SupplierConfirmationPage	DisplayError
.....	.....	.....	.....
.....	.....	.....	.....

Company A : Action Processor Mapping Table (AFTER)

Source	ActionProcessorName	Destination Page	ErrorPage
GenericMakeSale	CompanyA_InvoiceProcessor	GenericPrintInvoicePage	DisplayError
GenericOrderPage	GenericOrderProcessor	OrderConfirmationPage	DisplayError
CompanyASupplierPage	GenericSupplierProcessor	SupplierConfirmationPage	DisplayError
.....	.....	.....	.....
.....	.....	.....	.....

Figure 4.12: Action processor customization example

For example, a Company A runs its business on this proposed W2ASVB model framework by using all generic user interfaces and all generic action processors. At some stage, Company A asks for customization of its GenericInvoiceProcessor. In this case, the service provider needs to perform the following steps:

- Creates a new action processor class called CompanyA_InvoiceProcessor and customize it according to Company A's requirements.
- Compiles the code of this new action processor
- Plug-in this new action processor in the system by replacing the word GenericInvoiceProcessor with CompanyA_InvoiceProcessor in Company A's profile inside the meta-data module by the controller module.
- And finally, instructs the profile factory to refresh Company A's action processor mapping.

Figure 4.12 reflects the changes in Company A's action processor mapping before and after customization.

4.5.2 Customization of user interface

For instance, if at some point Company A decides to define its own sales invoice format rather than using the default, then this customization of the user interface will require the service provider to make the following changes:

- Create a new object and name it CompanyA_GenericMakeSale.
- Customize the sales invoice on the new object, where required.

- Assign the new object into the system by replacing the word. GenericMakeSale with CompanyA_MakeSale in Company A's profile inside the meta-data module by the controller.
- Update the existing user interface, mapping details of the make sale page inside the meta-data module with a new one. This change ensures that the system will open the new make sale page next time.
- Instruct the profile factory to refresh its action processing mapping mechanism.

Figure 4.13 reflects the changes in Company A's action processor mapping mechanism before and after customization. Since sales' invoice is a printable object, the service provider can customize the corresponding print invoice page in the same way. This is illustrated in the Figure 4.13.

Company A : Action Processor Mapping Table (BEFORE)

Source	ActionProcessorName	Destination Page	ErrorPage
GenericMakeSale	CompanyA_InvoiceProcessor	GenericPrintInvoicePage	DisplayError
GenericOrderPage	GenericOrderProcessor	OrderConfirmationPage	DisplayError
CompanyASupplierPage	GenericSupplierProcessor	SupplierConfirmationPage	DisplayError
.....	.....	.....	.....
.....	.....	.....	.....

Company A : Action Processor Mapping Table (AFTER)

Source	ActionProcessorName	Destination Page	ErrorPage
CompanyA_MakeSale	CompanyA_InvoiceProcessor	CompanyA_PrintInvoicePage	DisplayError
GenericOrderPage	GenericOrderProcessor	OrderConfirmationPage	DisplayError
CompanyASupplierPage	GenericSupplierProcessor	SupplierConfirmationPage	DisplayError
.....	.....	.....	.....
.....	.....	.....	.....

Figure 4.13: User Interface Customization Example

By adopting this approach, the system can provide a customization facility for both user interfaces and action processors for any company on demand at low cost. As a point of interest, the service provider needs to make no changes in the core system design to perform this customization.

4.6 System Interactions Diagrams

A better understanding of the overall component structure of W2ASVB has been developed. However, no discussion has been provided on how different classes of these components interact with each other to perform certain functionalities.

To understand the core functionality of the system, one needs to understand the following three system life-cycle interactions:

- Interactions between classes and objects that happen at system initialization time.
- Interactions between classes and objects that happen when employee (i.e. user) login to the system.
- Interactions between classes and objects that happen when employee sends a request to the system for executing business workflows (i.e. action processors).

The following sub-sections contain UML sequence diagrams to elaborate further on the above-mentioned interactions.

4.6.1 System Initialization Sequence Diagram

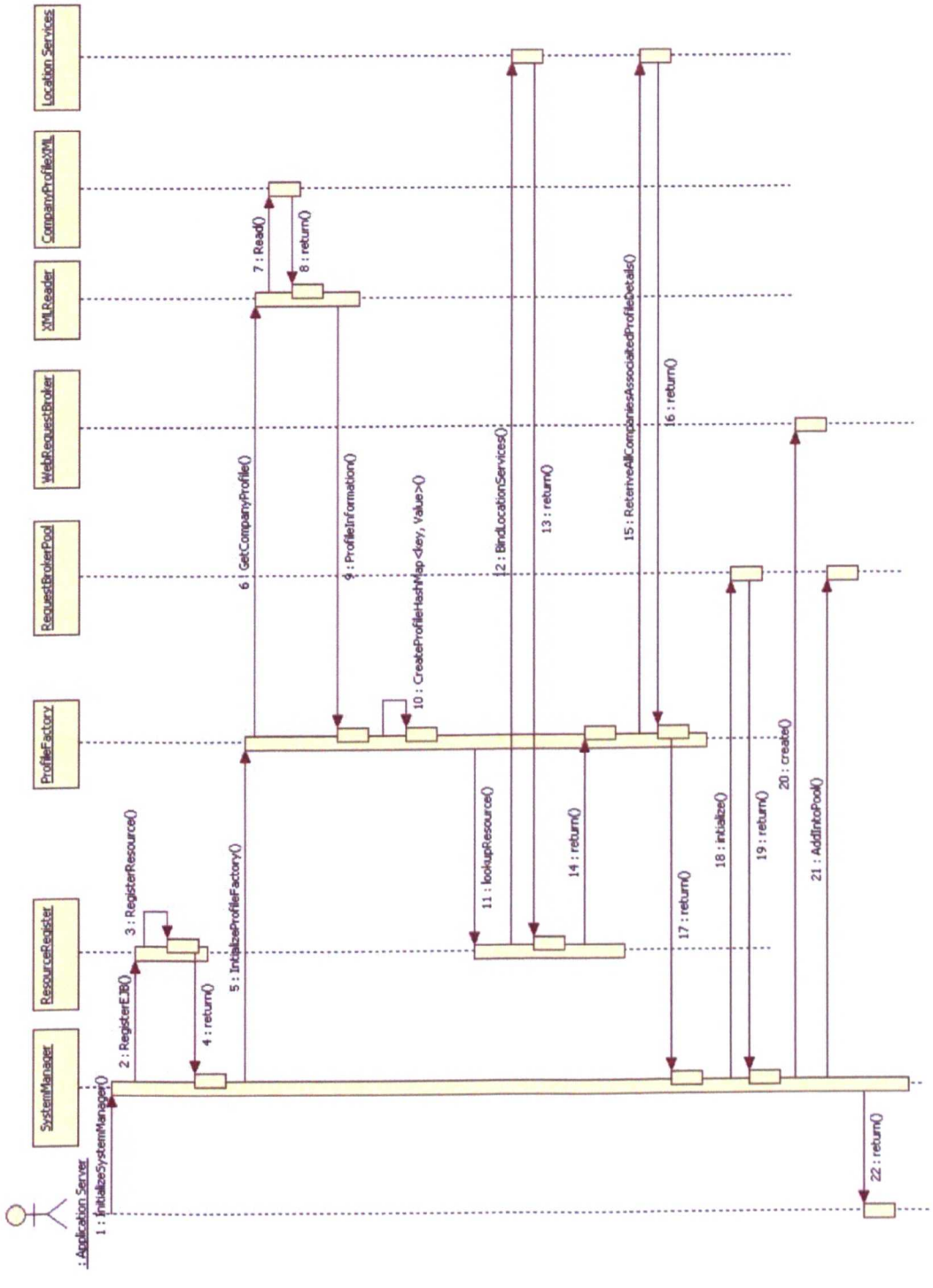


Figure 4.14: System initialization sequence diagram

4.6.2 Employee Login Sequence Diagram

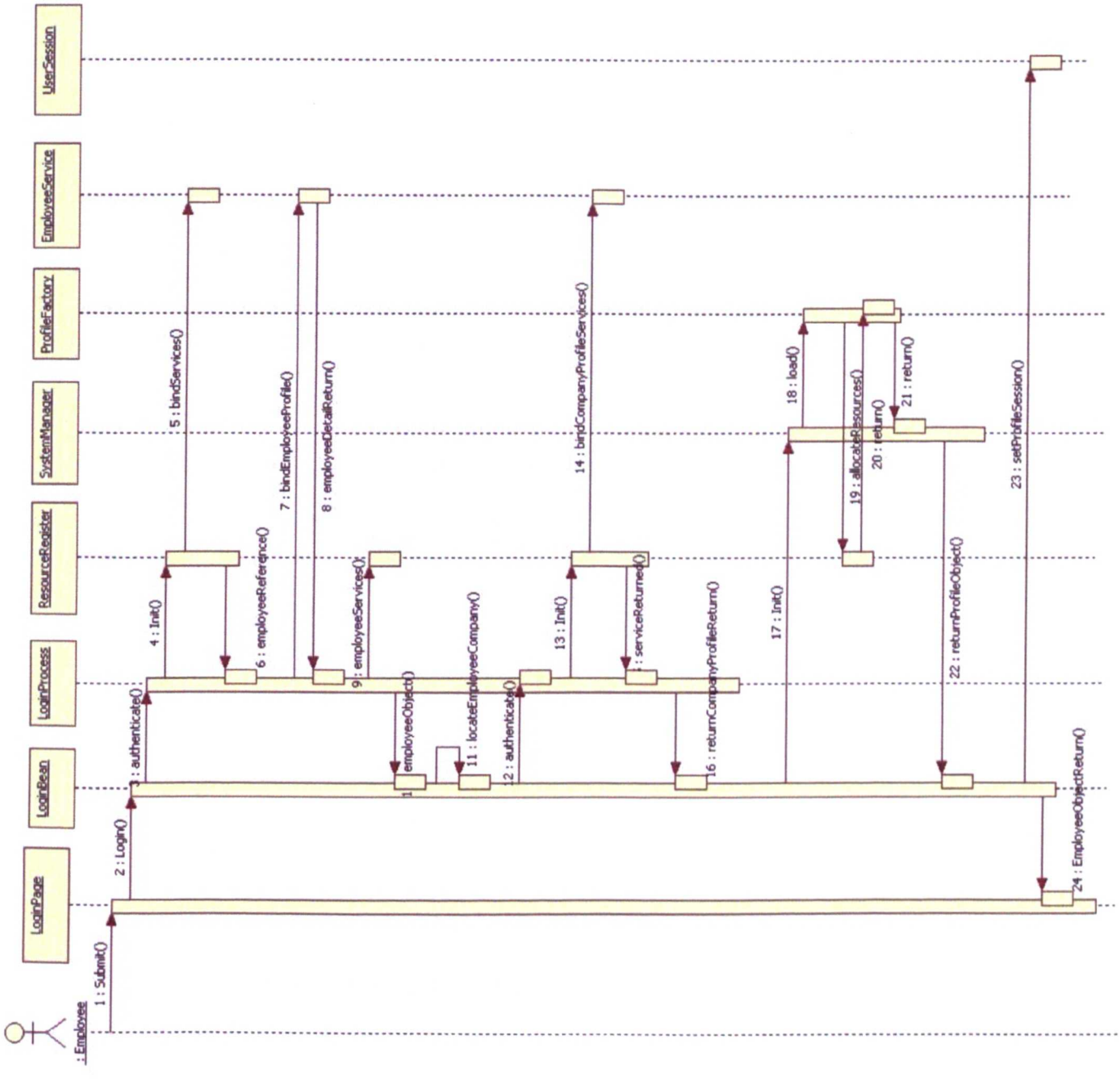


Figure 4.15: Employee login sequence diagram

4.6.3 Request Processing Sequence Diagram

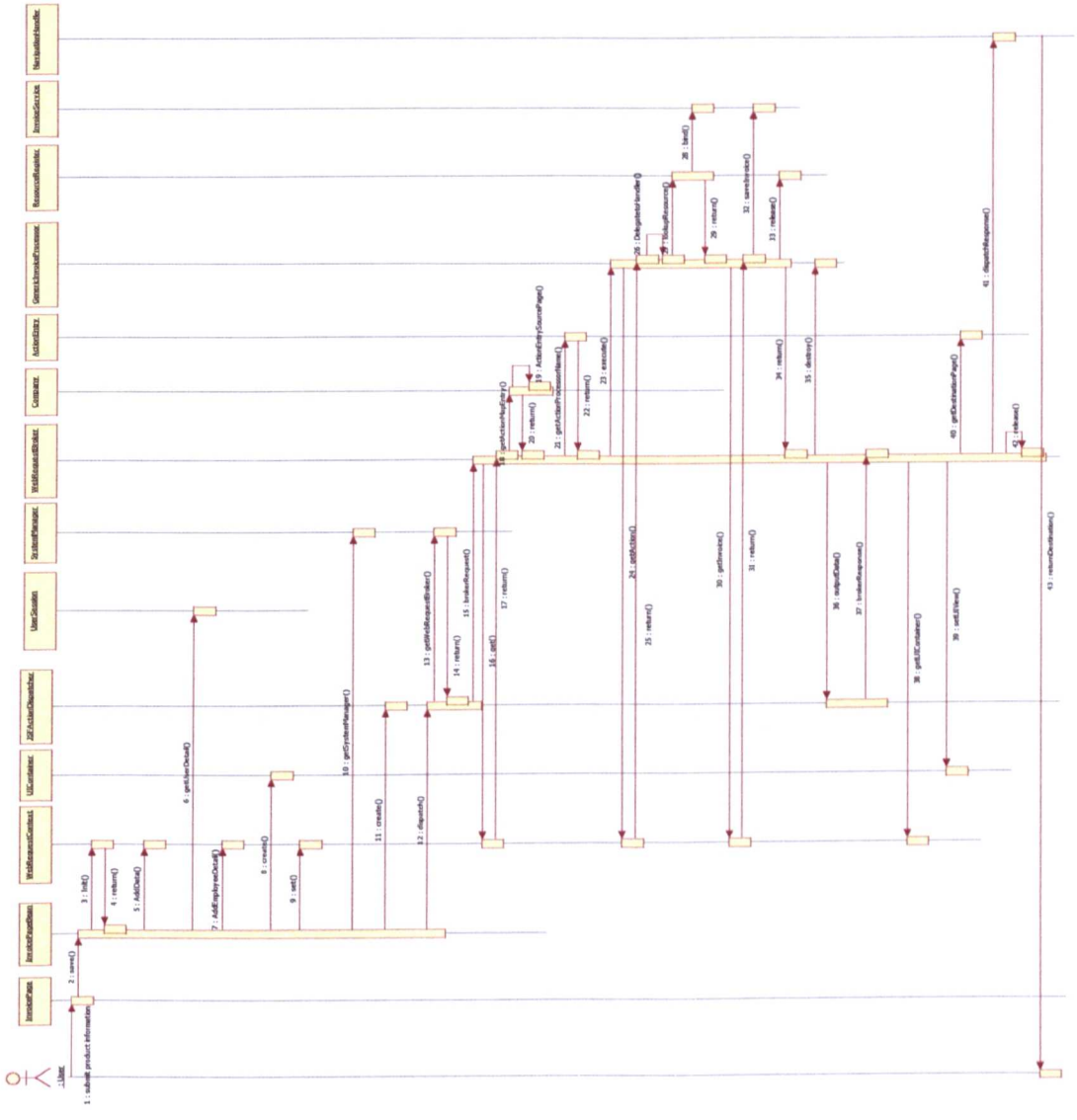


Figure 4.16: Request processing sequence diagram - Adding new invoice in the database

5. A W2ASVB framework implementation

5.1 Introduction

This section provides the implementation details of the W2ASVB framework. The correct implementation of any framework design facilitates the validation and verification of the framework and other design parameters. The section first presents a brief discussion about the chosen implementation language and technology. This is followed by a discussion of pseudo-code snippet from different parts of the implemented system along with the details of their execution.

5.2 Implementation Technology

As discussed in chapters 1 and 3, the W2ASVB model framework is using a core platform, which is implemented in Java using EJB technology. The choice of the language for the implementation of the system is Java. Moreover, the implementation of the proposed model framework in Java makes integration of different components of the system seamless and elegant within the existing core platform. As an additional benefit of this choice, no coding or integration related technology is required to develop collaboration mechanism between the components of W2ASVB and the core platform.

Whenever any web system goes under development, the selection of appropriate front-end user-interface frameworks plays a vital role. The careful selection of such a framework is critical for the success of the project. J2EE offers different UI frameworks for the development of a web system. Amongst them, JSF (Java Server Faces), Struts and Tiles are very good technologies. The application of these frameworks on a web system is entirely dependent on the type of the system in production. For example, Struts is known for its controller-based architecture whereas Tiles is known for changing corporate layouts in a seamless

manner. Apart from these two, JSF has a developed user interface framework designed by Sun Microsystems and is quickly getting closer to become an industry standard. Taking into consideration the popularity of the JSF framework and its acceptance as an industry standard; JSF has been adopted as a user interface framework for W2ASVB. More information about JSF can be found at (JSF, 2007).

5.3 Implementation Details

This section provides details on how certain parts of the design are translated into working code. Only the key pseudo-code snippets have been extracted from the respective classes and explained. It should be noted here that web systems built on the JSF framework, contain code that is distributed across different parts of the system. Hence, the complete code could not be understood unless its related parts are explained.

5.3.1 System Manager

init() is a method of the SystemManager module that is called when the system manager is initialized by the application server. On execution, this function first initializes the ResourceRegister, and then initializes the ProfileFactory followed by the initialization of the request broker pool (according to the given POOL_SIZE). This pool is then filled with web request brokers. POOL_SIZE is a variable that leads the system manager to decide about the size of the request broker pool.

The following pseudo-code presents the flow of control inside the *init()* method of the SystemManager.java file.

Start of Init()

1. *Initialize ResourceRegister*
2. *Initialize ProfileFactory*
3. *Initialize Request Broker Pool size with given POOL_SIZE*
4. *For i=0 to POOL_SIZE*

   4.1 Create new Web Request Broker

   4.2 Add newly created Web Request Broker in the Request Brokers pool

   [End of for loop]

For the complete code listing see Appendix (Section 10.1.1).

5.3.2 Profile Factory

The *init()* method of the ProfileFactory module is responsible for performing this task. It is called by the system manager module at the time when the manager becomes initialized. This method is responsible for creating a company's objects along with the initialization of their respective action-processor mapping tables, admin-menu mapping tables and sales-person mapping tables in memory. Moreover, this method is also responsible for creating the location objects of the corresponding companies and then putting them one by one in the <key, value> type collection object, where the key part represents the IP address of a location and the value part represents the location object associated with that IP address.

The following pseudo-code presents the flow of control inside the init() method of the ProfileFactory.java file.

Start of Init()

1. Declare <KEY, VALUE> type LocationProfiles Collection

2. Declare CompanyList collection

3. Load all the companies in CompanyList from CompanyProfiles.XML file

4. Loop through each Company in CompanyList

4.1. Fill up the Company Object with company information using core platform

4.2. Read the Action-Processing-Mapping table from CompanyProfiles.XML File

4.3. Store the Action-Processing-Mapping Table in Company object

4.4. Read the Admin-Menu-Mapping table from CompanyProfiles.XML

4.5. Store the Admin-Menu-Mapping table in Company object

4.6. Read the SalesPerson-Menu-Mapping table from CompanyProfiles.XML

4.7. Store the SalesPerson-Menu-Mapping table in Company object

4.8. Retrieve all the locations of Company from CompanyProfiles.XML file

4.9. Loop through each Location of Company

4.9.1. Create Location object

4.9.2. Fill up Location object with required information using core platform

4.9.3. Store reference of Company object in Location object

4.9.4. Put Location object in LocationProfiles collection with Location IP address as KEY and Location object as VALUE

End of Inner Loop

End of Outer Loop

End of Init()

For a complete code listing see Appendix (Section 10.1.2).

5.3.3 Session profile

The login() method of the Login module is responsible for performing this job. It is called when any employee of any company attempts to login to the system from any location. It validates the employee credentials and location access rights using the core platform's security service, and on successful authentication of both employee and location details, it passes the location IP address to the system manager to retrieve the associated location object. The system manager in turn passes this request to the profile factory to locate the location object associated with the given IP address. This location object holds the reference of the associated company object which in turn holds the action-processors, sales-menu and admin-menu mapping tables. On return from the call from the system manager, this method stores the location object in the user session profile and from this point onwards this user session profile serves as a reference document for the system to route future requests from this employee.

The following pseudo code presents the flow of control inside the login() method of the Login.java file.

Start of Login(UserId, Password, Role)

1. Create UserSessionProfile object

2. Create LoginProcessor

3. Authenticate employee credentials using LoginProcessor

3.1. If authenticated, then

3.1.1. Retrieve the employee details from core platform

3.1.2. Store the employee details in UserSessionProfile object

3.2. Else

3.2.1. Return un-authorized user access error message to user

[End of if-else]

4. Retrieve IP address of location of access

5. Authenticate location of access using LoginProcessor

5.1. If location authenticated, then

5.1.1. Retrieve Location object associated with IPAddress using System Manager

5.1.2. Store Location object in UserSessionProfile

5.2. Else

5.2.1. Return un-authorized location access error message to user

[End of if-else]

6. Destroy LoginProcessor

7. If employee role is Company Admin, then

7.1. Show Company Admin Desktop

8. Else

8.1. Show Sales Person Desktop

[End of if-else]

End of Login()

For complete code listing see Appendix (Section 10.1.4).

5.3.4 Web request broker

The getWebRequestBroker() method of the System Manager module is responsible for performing this task. It is called by the JSFActionDispatcher class when any request comes in to the system for processing. On receiving a call from the JSFActionDispatcher, the system manager iterates through the request broker pool to find a free web request broker. As soon as the system manager locates any free web request broker it sets its status to busy and returns its reference back to the JSFActionDispatcher for further processing.

The following pseudo-code presents the flow of control inside the getWebRequestBroker() method of SystemManager.java file.

```
Start of getWebRequestBroker()
    1. Do while there are WebRequestBrokers in request broker pool
        1.1. Pick up WebRequestBroker from Pool
        1.2. If WebRequestBroker is busy, then
            1.2.1.  Continue loop
        1.3. Else
            1.3.1.  Set WebRequestBroker busy flag to true
            1.3.2.  Break and Return WebRequestBroker
        [End of if-else]
    [End of While loop]
End of getWebRequestBroker()
```

For complete code listing see Appendix (Section 10.1.6).

5.3.5 Action Processor

The brokerRequest() method of the Web Request Broker module is responsible for creating a user-profile session at the time of login. It is called by the

JSFActionDispatcher module after the allocation of free web request broker. Moreover, the JSFActionDispatcher passes the reference of the web request context object to this method as well. This web request context object holds the details such as details of an employee who sends the request, location details of the location from which request gets generated and a source name of the page. This method loads the corresponding action processor with the help of source page name and action processor mapping table. Each line of the action processor mapping table is implemented as a single instance of an action entry or, in other words each <Action> tag of XML meta-data file is translated into one action entry. In memory, the action-processor mapping table is a collection of action entry objects. At the end, control passes back to the JSFActionDispatcher, which then calls the brokerResponse() to send the response back to the appropriate user interface.

The following pseudo-code presents the flow of control inside the brokerRequest() method of the WebRequestBroker.java file.

Start of brokerRequest(WebRequestContext)

1. Read Source Page name from WebRequestContext

2. Retrieve Location Profile from WebRequestContext

3. Lookup ActionEntry corresponding to Source Page from UserSessionProfile

4. Read Action Processor name from Action Entry

5. Load and execute corresponding Action Processor

5.1. If executed successfully, then

5.1.1. Store output in UIView object

5.2. Else

5.2.1. Store exception in AppException object

[End of if-else]

6. Unload Action Processor

End of brokerRequest()

For complete code listing see Appendix (Section 10.1.7).

5.3.6 Mapping output data to the user interface

The brokerResponse() method of the Web Request Broker module is responsible for routing back the output data from the action processor to the appropriate user interface. It gets called by the JSFActionDispatcher when the brokerRequest() method finishes its execution. Its main responsibility is to route the response back to the appropriate destination page. This method creates the UIContainer and stores the UIView within it. It then reads the action entry to find the destination page. It then dispatches the response back to the destination page. At the end, the JSFActionDispatcher calls the release() method of the web request broker to make it available for the next processing request.

The following pseudo-code describes the flow of control inside the brokerResponse() method of the WebRequestBroker.java file.

Start of brokerResponse(WebRequestContext, WebResponseContext)

1. Create UIContainer

2. Reads AppException from WebRequestContext

2.1. If AppException is null, then

2.1.1. Store UIView in UIContainer

2.2. Else

2.2.1. Store AppException in UIContainer

[End of if-else]

3. Store UIContainer in WebRequestContext

4. Read Destination Page name from ActionEntry

5. Dispatch response to the Destination Page

End of brokerRequest()

5.4 System Outputs

```
SystemManager --> init() --> Start
SystemManager --> init() --> Initializing EJB Platform
ResourceRegisterar --> registerEJBS() --> Start
ResourceRegisterar --> registerEJBS() --> End
SystemManager --> init() --> EJB Platform Initialization complete.
SystemManager --> init() --> Initializing ProfileFactory.
ProfileFactory() --> getInstance() --> Start
ProfileFactory --> init() --> Start
XMLReader --> getInstance() --> Start
XMLReader --> getInstance() --> End
XMLReader --> getCompaniesList() --> Start
Root node of XML document is: Company-Profiles
Total no of Company Profiles in XML file : 2
Company ID: 61
URL : www.ca-electronics.com
Total number of registered Locations for this Company : 3
Location IP-Address=217.35.95.208 logopath=resources/logodir/CompanyName1-LocationName1.gif
Location IP-Address=217.45.161.252 logopath=resources/logodir/CompanyName1-LocationName2.gif
Location IP-Address=217.35.95.138 logopath=resources/logodir/CompanyName1-LocationName2.gif
Total number of Action-Entry : 6
[Action-Entry source->genericMakeSale.jsp dest->showGenericPrintInvoice actionProcessor-
>generic.InvoiceProcessor errorpage->showGenericMakeSaleError]
[Action-Entry source->genericStockCheck.jsp dest->NULL actionProcessor-
>generic.stock.StockCheckProcessor errorpage->NULL]
[Action-Entry source->genericSupplierRelation.jsp dest->NULL actionProcessor-
>generic.supplier.SupplierProcessor errorpage->showSupplierRelationErrorPage]
[Action-Entry source->GenericAddProduct.jsp dest->messages actionProcessor->ProductProcessor
errorpage->error]
[Action-Entry source->GenericAddCustomer.jsp dest->messages actionProcessor->CustomerProcessor
errorpage->error]
[Action-Entry source->GenericAddSupplier.jsp dest->messages actionProcessor->SupplierProcessor
errorpage->error]
Total number of Admin Menu links: 11
[Menu name->desktop navigation-case->showGenericAdminDesktop]
[Menu name->recorddelivery navigation-case->/samand/faces/genericRecordDelivery.jsp]
[Menu name->reprintlabels navigation-case->/samand/faces/genericReprintLabels.jsp]
[Menu name->printsalesreports navigation-case->/samand/faces/genericPrintSalesReports.jsp]
[Menu name->printpurchasereports navigation-case->/samand/faces/genericPrintPurchaseReports.jsp]
[Menu name->printstockreports navigation-case->/samand/faces/genericPrintStockReports.jsp]
[Menu name->printattendancereports navigation-case->/samand/faces/genericPrintAttendanceReport.jsp]
[Menu name->addsupplier navigation-case->/samand/faces/genericSupplierRelation.jsp]
[Menu name->updatesupplier navigation-case->/samand/faces/genericSupplierRelation.jsp]
[Menu name->closetill navigation-case->/samand/faces/genericCloseTill.jsp]
[Menu name->resetstaffpassword navigation-case->/samand/faces/genericResetStaffPassword.jsp]
Total number of Salesperson Menu links: 7
[Menu name->desktop navigation-case->showGenericSalesPersonDesktop
[Menu name->stockcheck navigation-case->/samand/faces/genericStockCheck.jsp
[Menu name->makesale navigation-case->/samand/faces/genericMakeSale.jsp
[Menu name->managesupplier navigation-case->/samand/faces/genericSupplierRelation.jsp
[Menu name->viewperformance navigation-case->/samand/faces/genericPerformance.jsp
[Menu name->movestock navigation-case->/samand/faces/genericMoveStock.jsp
[Menu name->supplierrelations navigation-case->/samand/faces/genericSupplierRelations.jsp
----------------------
```

```
Company ID: 1
URL : www.N-Genius.com
Total number of registered Locations for this Company : 1
Location IP-Address=217.41.27.94 logopath=resources/logodir/CompanyName2-LocationName1.gif
Total number of Action-Entry : 6
[Action-Entry source->genericMakeSale.jsp dest->showGenericPrintInvoice actionProcessor-
>generic.InvoiceProcessor errorpage->showGenericMakeSaleError]
[Action-Entry source->genericStockCheck.jsp dest->NULL actionProcessor-
>generic.stock.StockCheckProcessor errorpage->NULL]
[Action-Entry source->genericSupplierRelation.jsp dest->NULL actionProcessor-
>generic.supplier.SupplierProcessor errorpage->showSupplierRelationErrorPage]
[Action-Entry source->GenericAddProduct.jsp dest->messages actionProcessor->ProductProcessor
errorpage->error]
[Action-Entry source->GenericAddCustomer.jsp dest->messages actionProcessor->CustomerProcessor
errorpage->error]
[Action-Entry source->GenericAddSupplier.jsp dest->messages actionProcessor->SupplierProcessor
errorpage->error]
Total number of Admin Menu links: 11
[Menu name->desktop navigation-case->showGenericAdminDesktop]
[Menu name->recorddelivery navigation-case->/samand/faces/genericRecordDelivery.jsp]
[Menu name->reprintlabels navigation-case->/samand/faces/genericReprintLabels.jsp]
[Menu name->printsalesreports navigation-case->/samand/faces/genericPrintSalesReports.jsp]
[Menu name->printpurchasereports navigation-case->/samand/faces/genericPrintPurchaseReports.jsp]
[Menu name->printstockreports navigation-case->/samand/faces/genericPrintStockReports.jsp]
[Menu name->printattendancereports navigation-case-
>/samand/faces/genericPrintAttendanceReport.jsp]
[Menu name->addsupplier navigation-case->/samand/faces/genericSupplierRelation.jsp]
[Menu name->updatesupplier navigation-case->/samand/faces/genericSupplierRelation.jsp]
[Menu name->closetill navigation-case->/samand/faces/genericCloseTill.jsp]
[Menu name->resetstaffpassword navigation-case->/samand/faces/genericResetStaffPassword.jsp]
Total number of Salesperson Menu links: 7
[Menu name->desktop navigation-case->showGenericSalesPersonDesktop
[Menu name->stockcheck navigation-case->/samand/faces/genericStockCheck.jsp
[Menu name->makesale navigation-case->/samand/faces/genericMakeSale.jsp
[Menu name->managesupplier navigation-case->/samand/faces/genericSupplierRelation.jsp
[Menu name->viewperformance navigation-case->/samand/faces/genericPerformance.jsp
[Menu name->movestock navigation-case->/samand/faces/genericMoveStock.jsp
[Menu name->supplierrelations navigation-case->/samand/faces/genericSupplierRelations.jsp
----------------------
XMLReader --> getCompaniesList() --> End
ProfileFactory --> init() --> locationHome.findByIP(217.35.95.208) = 1
ProfileFactory --> init() --> locationHome.findByIP(217.45.161.252) = 6
ProfileFactory --> init() --> locationHome.findByIP(217.35.95.138) = 1
ProfileFactory --> init() --> locationHome.findByIP(217.41.27.94) = 1
ProfileFactory --> init() --> End
ProfileFactory() --> getInstance() --> End
SystemManager --> init() --> ProfileFactory initialization complete.
```

Figure 5.1: System Initialization - Loading process of company profiles from CompanyProfile.XML file

```
SystemManager --> init() --> About to call initRequestBrokersPool().
SystemManager --> initRequestBrokersPool() --> Start
SystemManager --> initRequestBrokersPool() --> POOL SIZE set to->10
WebRequestBroker --> init() --> Start
WebRequestBroker --> init() --> Initialized with BUSY status :false
WebRequestBroker --> init() --> End
SystemManager --> initRequestBrokersPool() --> WebRequestBroker
[samand.broker.WebRequestBroker@8cb0d2] added in pool at position->0
WebRequestBroker --> init() --> Start
WebRequestBroker --> init() --> Initialized with BUSY status:false
WebRequestBroker --> init() --> End
SystemManager --> initRequestBrokersPool() --> WebRequestBroker
[samand.broker.WebRequestBroker@acdd8a] added in pool at position->1
WebRequestBroker --> init() --> Start
WebRequestBroker --> init() --> Initialized with BUSY status:false
WebRequestBroker --> init() --> End
SystemManager --> initRequestBrokersPool() --> WebRequestBroker
[samand.broker.WebRequestBroker@1049065] added in pool at position->2
WebRequestBroker --> init() --> Start
WebRequestBroker --> init() --> Initialized with BUSY status:false
WebRequestBroker --> init() --> End
SystemManager --> initRequestBrokersPool() --> WebRequestBroker
[samand.broker.WebRequestBroker@a3e8d9] added in pool at position->3
WebRequestBroker --> init() --> Start
WebRequestBroker --> init() --> Initialized with BUSY status:false
WebRequestBroker --> init() --> End
SystemManager --> initRequestBrokersPool() --> WebRequestBroker
[samand.broker.WebRequestBroker@10bfee3] added in pool at position->4
WebRequestBroker --> init() --> Start
WebRequestBroker --> init() --> Initialized with BUSY status:false
WebRequestBroker --> init() --> End
SystemManager --> initRequestBrokersPool() --> WebRequestBroker
[samand.broker.WebRequestBroker@8aedb8] added in pool at position->5
WebRequestBroker --> init() --> Start
WebRequestBroker --> init() --> Initialized with BUSY status:false
WebRequestBroker --> init() --> End
SystemManager --> initRequestBrokersPool() --> WebRequestBroker
[samand.broker.WebRequestBroker@fd05c7] added in pool at position->6
WebRequestBroker --> init() --> Start
WebRequestBroker --> init() --> Initialized with BUSY status:false
WebRequestBroker --> init() --> End
SystemManager --> initRequestBrokersPool() --> WebRequestBroker
[samand.broker.WebRequestBroker@1868cf3] added in pool at position->7
WebRequestBroker --> init() --> Start
WebRequestBroker --> init() --> Initialized with BUSY status:false
WebRequestBroker --> init() --> End
SystemManager --> initRequestBrokersPool() --> WebRequestBroker
[samand.broker.WebRequestBroker@8fb08f] added in pool at position->8
WebRequestBroker --> init() --> Start
WebRequestBroker --> init() --> Initialized with BUSY status:false
WebRequestBroker --> init() --> End
SystemManager --> initRequestBrokersPool() --> WebRequestBroker
[samand.broker.WebRequestBroker@190973e] added in pool at position->9
SystemManager --> initRequestBrokersPool() --> End
SystemManager --> init() --> End
```

Figure 5.2: System Initialization - Initialization of request broker pool

6. A W2ASVB framework validation

This chapter presents the results of testing conducted on W2ASVB. The primary purpose of the testing was to validate the system functionalities by comparing them with other software. This chapter first describes the environment in which testing was conducted. This is followed by the list of the test cases that were developed to perform testing and finally, it presents the detailed description of test results in tabular form. The presentation format for test results description has been tailored from IEEE 829:1998: Standard for Software Test Documentation and the testing approach is black box testing.

6.1 Environment

The table below contains the description of hardware and software environment under which the test cases were executed:

6.1.1 Hardware Environment

CPU	Intel 1.5 GHz Centrino Mobile Technology
RAM	1.5 GB
Hard Disk	60 GB
Display Card	AGP Graphics card with 32 bit support
Monitor	SVGA
Keyboard	Standard 101 keys
Mouse	Standard

6.1.2 Software Environment

Operating System	Windows Vista Home Edition
Application Server	Sun One Application Server 9 and IIS

IDE	Netbeans 5.5.1 with Visual Web Pack installed
Internet Browser	Mozilla Firefox v 2.0.0.6

6.2 Validation process

Validation of the W2ASVB framework is carried out by comparing its execution time, average cost per user and scalability with the ZohoCRM and Microsoft Dynamics. The reason for selecting the ZohoCRM and Microsoft Dynamics for validation is that both consist of several features such as partial customization, generic workflow management mechanism etc. similar to W2ASVB and also both have their own software validation platform for providing data such as instruction execution time etc.

6.2.1 Validation testing cases

Test Case	Description
Execution Time	Executing the "Make Sale" activity process in ZohoCRM, Microsoft Dynamic and W2ASVB and then comparing the results.
Average Cost per User	Comparing the initial and customization cost of ZohoCRM, Microsoft Dynamic and W2ASVB framework.
Scalability	Using the Logic versus physical tiers scalability validation method to compare the systems.

6.2.2 Execution time

The execution time of a computational task is the length of time the task takes to execute a specific platform. Knowing the execution time is of a major importance for the analysis of real-time system performance.

The execution time is validated by running the set of the instructions under the "Make Sale" activity as shown below. Subsequently, the process and the execution time of the entire request are stored within the Profilefactory object XML collections with the key and the associated values. These steps will help to validate the model framework execution time as it is considered as a major factor of any new e-business model. The execution time outcome comparison with existing e-business frameworks along with execution process of the entire selected and proposed model framework is presented in the next few sub sections. The execution time of the different frameworks can be calculated by performing the following steps.

Identifier	T1
Setup	User is logged on to the system.
Description	To test the make sale functionality of the system. Also, to test the request routing from generic user interface to generic action processor by web request broker.
Procedure	User clicks on the 'Make Sale' button.

	TC 1.1 User enters the customer items codes TC 1.2 User enters the payments details TC 1.3 User enters the customer details TC 1.4 User presses 'save' button
Modules Executed	• genericMakeSale.jsp • genericMakeSale.java • genericItemSale.jsp • genericItemSale.java • genericPaymentScan.jsp • genericPaymentScan.java • genericCustomerScan.jsp • genericCustomerScan.java • InvoinceBean.java • ResourceRegister.java • JSFActionDispatcher.java • SystemManager.java • WebRequestBroker.java

	• GenericInvoiceProcessor.java
Result	For TC 1.1 • System adds the given items on the sale invoice (see Figure 6.1). For TC 1.2 • System adds the given payments on the sale invoice (see Figure 6.2). For TC 1.3 • System registers the customer details in the system and shows confirmation on the sale invoice (see Figure 6.3). For TC 1.4 • New sale is recorded successfully in the system. • Relevant product quantity is updated. • System prints out the sale invoice (see Figure 6.4).

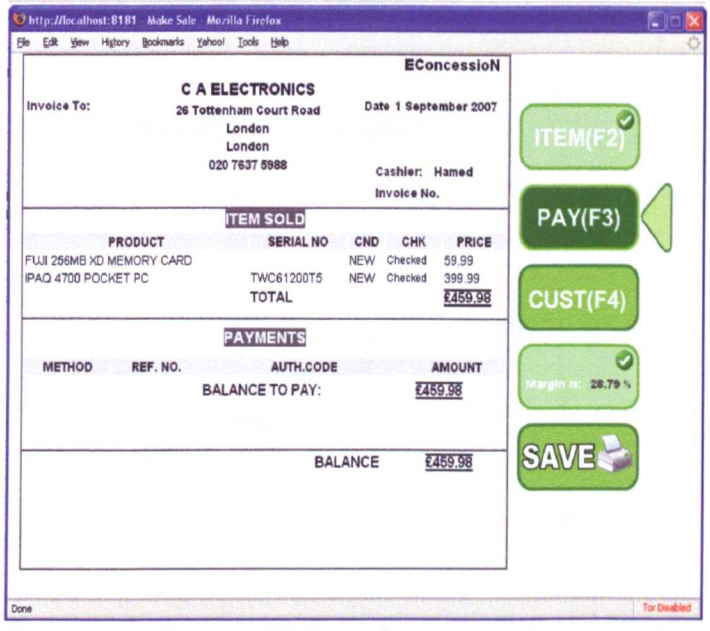


Figure 6.1: System confirms addition of customer items on the sale invoice

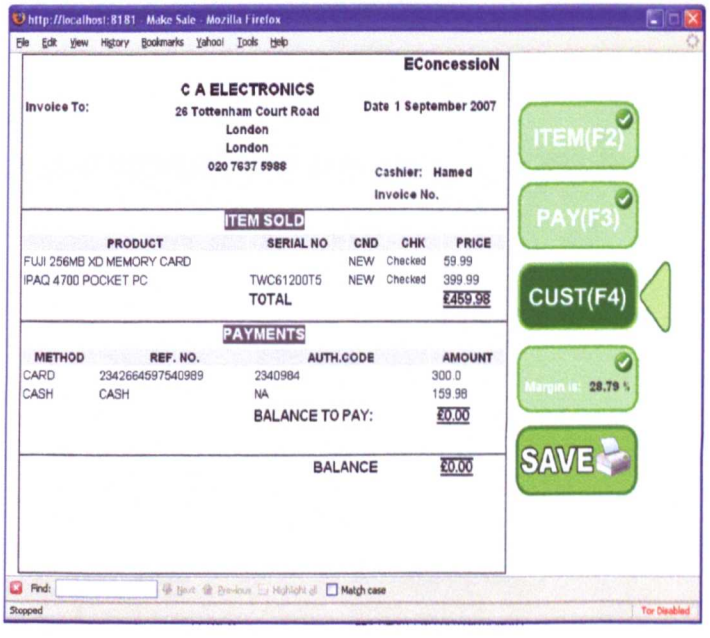


Figure 6.2: System confirms collection of customer payments on the sale invoice

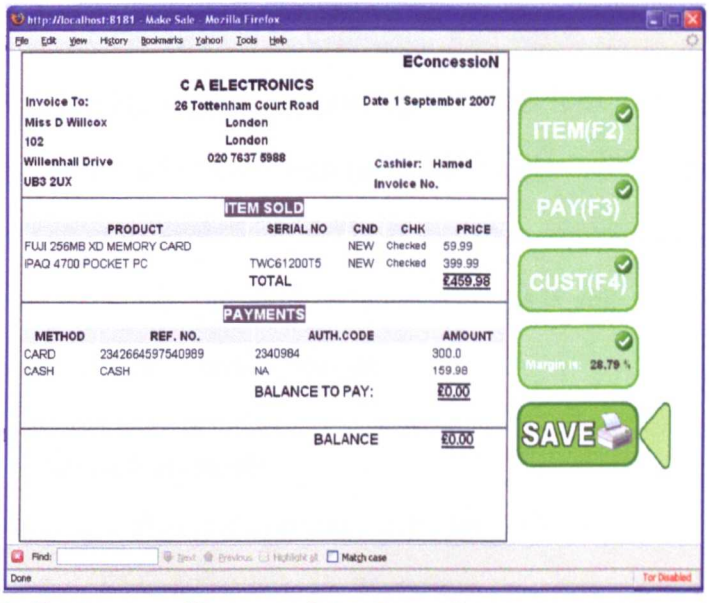


Figure 6.3: System confirms registration of a new customer on the sale invoice

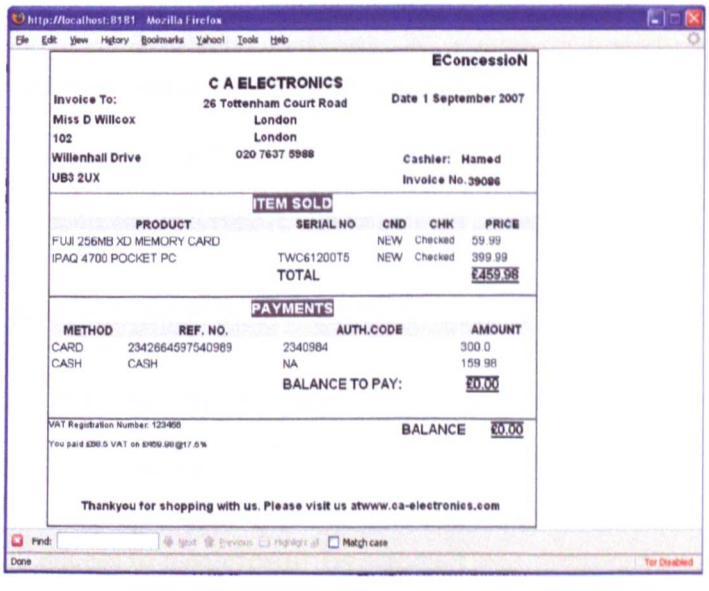


Figure 6.4: System shows the printable version of the sales invoice

6.2.2.1 Zoho CRM framework

ZohoCRM handles the user request by a conventional request management technique that is commonly used in traditional web-based systems. On arrival of a request, an application server automatically redirects it to a designated module for processing. At the end of the processing, the module redirects control back to the application server by specifying the response page address (e.g. confirmation or an error page). Finally the application server loads the response page and sends it to the user where it gets displayed.

6.2.2.2 Microsoft Dynamic

Microsoft Dynamics CRM user request processes are based on the Windows Workflow Foundation model. Windows Workflow Foundation runs a framework, a base library of activities, and default implementations of the runtime services. The Windows Workflow Foundation runtime engine manages process execution, and supports processes that can remain active for extended periods of time. It preserves the state of process execution during the request process and returns back the results to the user after processing. The steps for executing the validation process on the Microsoft Dynamics are given below.

- Install and open the SDK in Visual C# Express.
- Access to the tool's source code in the SDK\Tools\PluginDeveloper folder of the SDK installation.
- Network access to a Microsoft Dynamics CRM 4.0 server.
- Login in Microsoft Dynamics CRM system account.

- Execute the "Make Sale" action by using the SDK.
- Store the execution time in XML.

6.2.2.3 Proposed W2ASVB model framework

Any user request for data processing is divided into two parts – data and action part. The data part contains the *data* that requires processing, whereas the action part describes the required operation on the given data. Request brokers are a key concept in the proposed W2ASVB model framework. They are semi-autonomous objects which are completely equipped with all the capabilities to handle action and view management for user requests. In other words, they can be called middle agents that provide a mediation mechanism. They have been classified as semi-autonomous because they can complete the user's request without any assistance from other brokers but are bound to be controlled by managing authority. This managing authority is called the system manager.

On arrival of a user request, it gets queued up in waiting area. The system manager continuously checks the waiting area and as soon as a request arrives, it is allocated to one request broker. This allocated request broker moves the request from the waiting area into the processing area and starts analysis of the request header to find the source details, such as the name of user interface from which the request is being generated, and the action it requires. On identification of source and action, it starts searching to find the name of a matching service adapter in the user session profile. The user session profile is a profile carried by each user, and is initially allocated to each user by the system manager when the user first logs in.

6.2.2.4 Results

The results of the execution tasks are given below.

Table 6-1: Results of the execution task

Number of instructions	Zoho CRM	Microsoft CRM	W2ASVB model framework
100 instructions	0.25 seconds for processing	0.20 seconds for processing	0.24 seconds for processing
1000 instructions	0.45 seconds for processing	0.35 seconds for processing	0.38 seconds for processing
10000 instructions	2min 20 seconds for processing	1min 40 seconds for processing	1min 10 seconds for processing

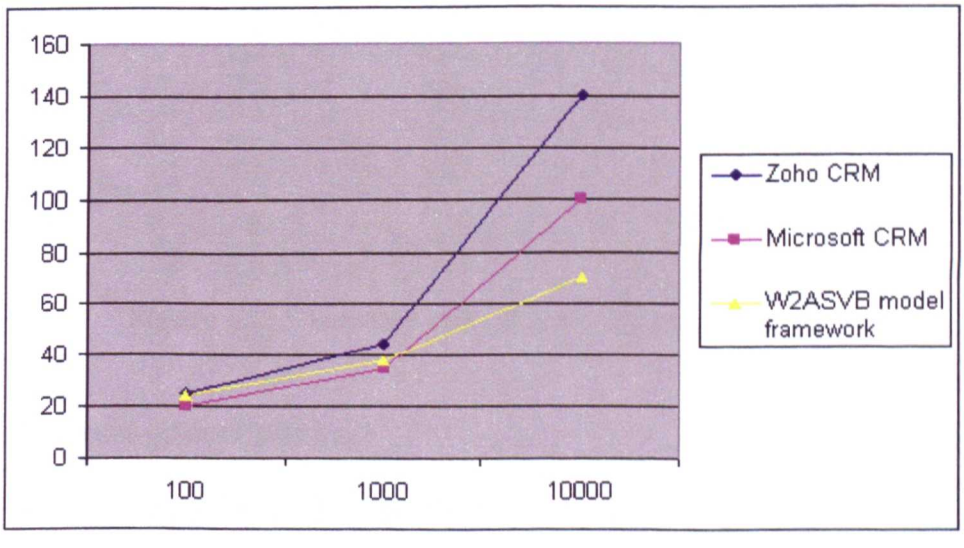
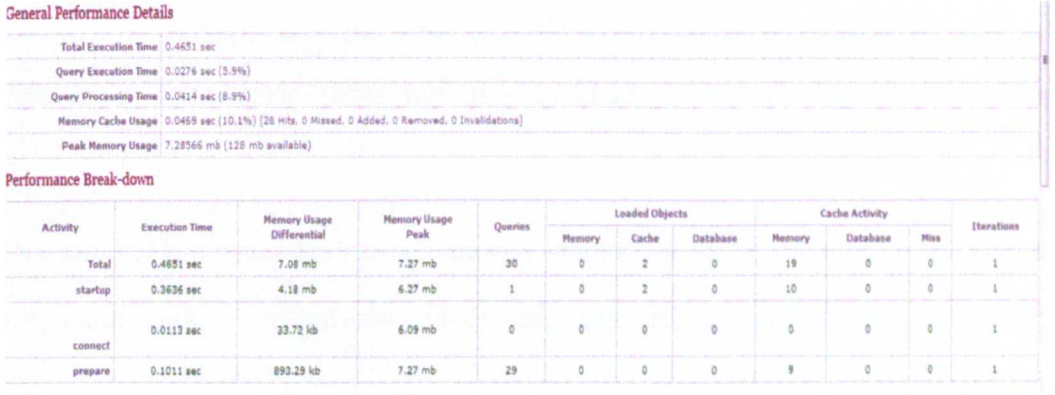


Figure 6.5: Results of the execution time validation

Figure 6.5 shows the execution time of the different frameworks on the provided number of instructions. Execution time is a sum of the different activities in each task as shown in Figures 6.6 and 6.7.



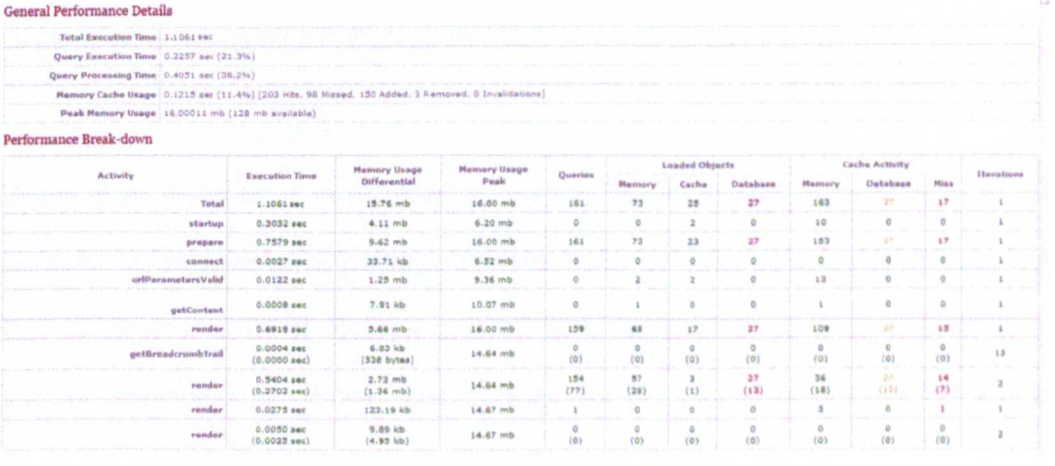
General Performance Details

Total Execution Time	0.4651 sec
Query Execution Time	0.0276 sec (5.9%)
Query Processing Time	0.0414 sec (8.9%)
Memory Cache Usage	0.0469 sec (10.1%) [26 Hits, 0 Missed, 0 Added, 0 Removed, 0 Invalidations]
Peak Memory Usage	7.28566 mb (128 mb available)

Performance Break-down

Activity	Execution Time	Memory Usage Differential	Memory Usage Peak	Queries	Loaded Objects			Cache Activity			Iterations
					Memory	Cache	Database	Memory	Database	Miss	
Total	0.4651 sec	7.08 mb	7.27 mb	30	0	2	0	19	0	0	1
startup	0.3636 sec	4.18 mb	6.27 mb	1	0	2	0	10	0	0	1
connect	0.0113 sec	33.72 kb	6.09 mb	0	0	0	0	0	0	0	1
prepare	0.1011 sec	893.29 kb	7.27 mb	29	0	0	0	9	0	0	1

Figure 6.6: Execution time of the ZohoCRM



General Performance Details

Total Execution Time	1.1061 sec
Query Execution Time	0.2257 sec (21.3%)
Query Processing Time	0.4051 sec (38.2%)
Memory Cache Usage	0.1215 sec (11.4%) [203 Hits, 98 Missed, 150 Added, 3 Removed, 0 Invalidations]
Peak Memory Usage	16.00011 mb (128 mb available)

Performance Break-down

Activity	Execution Time	Memory Usage Differential	Memory Usage Peak	Queries	Loaded Objects			Cache Activity			Iterations
					Memory	Cache	Database	Memory	Database	Miss	
Total	1.1061 sec	15.76 mb	16.00 mb	161	73	25	27	163	27	17	1
startup	0.3032 sec	4.11 mb	6.20 mb	0	0	2	0	10	0	0	1
prepare	0.7579 sec	9.62 mb	16.00 mb	161	73	23	27	153	27	17	1
connect	0.0027 sec	33.71 kb	6.32 mb	0	0	0	0	0	0	0	1
urlParametersValid	0.0122 sec	1.25 mb	9.36 mb	0	2	2	0	13	0	0	1
getContent	0.0008 sec	7.91 kb	10.07 mb	0	1	0	0	1	0	0	1
render	0.6818 sec	5.66 mb	16.00 mb	139	68	17	27	109	27	13	1
getBreadcrumbTrail	0.0004 sec (0.0000 sec)	6.03 kb (338 bytes)	14.64 mb	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	13
render	0.5404 sec (0.2702 sec)	2.73 mb (1.36 mb)	14.64 mb	154 (77)	57 (28)	3 (1)	27 (13)	36 (18)	27 (13)	14 (7)	2
render	0.0275 sec	123.19 kb	14.67 mb	1	0	0	0	3	0	1	1
render	0.0050 sec (0.0025 sec)	9.89 kb (4.95 kb)	14.67 mb	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	2

Figure 6.7: Execution time of the Microsoft Dynamic

6.2.3 Average cost per user

The average cost per user of the system depends upon the software, hardware and customisation costs of the system. The average cost per user of the system

implementation varies from one company to another, due to different hardware configuration and customisation. Each company has a number of different departments, and each department within an organization can be diverse, so it is imperative for software to be tailored to meet the unique requirements of its users. The average cost per user of ZohoCRM, Microsoft Dynamics and the W2ASVB framework under test is calculated by considering the following scenario.

We applied the framework to a company which has about 300 employees, and is organizationally divided into four departments, reflecting three different segments: financial, commercial, technical, and management. Each department has a different workflow on the same set of instructions; the management department needs live stock statistics while the financial department requires order history to manage the accounts etc.

6.2.3.1 ZohoCRM framework

ZohoCRM is available in different packages. The most suitable package for our scenario is the enterprise package and costs £13 per user per month. We need to customize this to fulfil the demands of each department, as currently all users get the same set of options. The average cost for each department's business logic implementation is approximately £10,000. It requires three developers to work about 10 full working days. This customisation is only possible from the ZohoCRM dedicated developer.

Table 6-2: Average cost per user of ZohoCRM

	Cost	Total cost
300 Users	£13 per user	3900 per month
4 Department customisation	10,000 per department	40,000 (Initial cost)

6.2.3.2 Microsoft Dynamics

Microsoft Dynamics CRM is available in different packages. The most suitable package for our scenario is the CRM online package as the other packages need a company-based server installation. Its cost is £23 per user per month. One needs customization to fulfil the demands of each department, as currently all users get the same set of options. The average cost for each department's business logic implementation is approximately £5,000. Microsoft's CRM provides several customization options to fulfil the different departmental requirements.

Table 6-3: Average cost per user of Microsoft Dynamics

	Cost	Total cost
300 User	£23 per user	£6900 per month
4 Department customisation	5,000 per department	20,000 (Initial cost)

6.2.3.3 Proposed W2ASVB model framework

Average cost of the W2ASVB framework is calculated by using the above-mentioned scenario. The approximate costing for the proposed model framework is given below:

Table 6-4: Average cost per user of W2ASVB model

	Cost	Total cost
300 Users	£2 per user (100MB data usage)	£600 per month
4 department customisation	App.£100 (It depends on the server configuration)	£100 (Initial setup cost)

6.2.3.4 Results

The results of the average cost per user analysis are given below;

Table 6-5: Results of the average cost per user

	First Month	Second Month	Third month
Zoho CRM	£43900	£3900	£3900
Microsoft Dynamic	£26900	£6900	£6900
W2ASVB model framework	£700	£ 600	£600

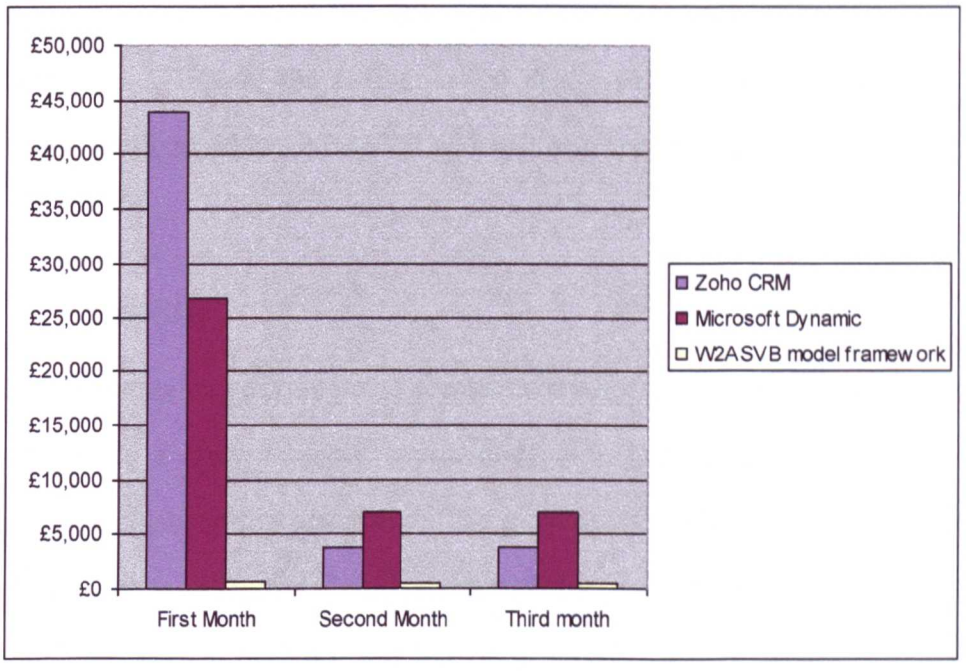


Figure 6.8: Results of the average cost per user validation

In the new proposed model framework customisation for the different departments is achieved by a customised business login for different departments by performing the following steps;

- Create a new action processor class called <<Company_Department1>> and customize it according to the Company's departmental requirements.
- Compile the code of this new action processor.
- Plug-in this new action processor in the system by replacing the word <<GenericProcessor>>with <<Company_Department1>> in the Company department profile in the XML meta-data file.

And finally, instruct the profile factory to refresh the Company action processor mapping table. Due to the customisation at the software level, it reduces the customisation cost of the W2ASVB and will also reduce the overall average cost per user.

6.2.4 Scalability

Careful planning and development are necessary for any framework development and to make a truly scalable application, it is important to rigorously and regularly validate it for scalability. The purpose of scalability validation is to identify major workloads and mitigate bottlenecks that can impede the scalability of the framework. There are number of different ways to perform scalability validation, such as cluster technologies, isolated transactional methods and business logic layer elimination etc. Logic versus physical tier scalability validation method has been used. In this method, all the layers of the framework such as the business logic layer and the data access layer are distributed and then tested by increasing the number of the users and their interfaces in the framework. The scalability of the ZohoCRM, Microsoft Dynamics and W2ASVB is performed by the following steps.

Identifier	T2
Setup	CompanyProfiles.xml file is present on designated path inside application server's hard disk.

Description	To test the scalability, initialization and organization of company profiles by Profile Factory from XML file, at user login.
Procedure	No explicit call is required to initiate this test-case; it is called internally by the system manager.
Modules Executed	• SystemManager.java • ProfileFactory.java • XMLReader.java • ResourceRegister.java
Result	• Profile factory successfully reads the company profiles from XML file. • Profile factory successfully initializes the action processor mapping tables for each company. • Profile factory successfully initializes the sales person menu mapping tables for each company. • Profile factory successfully initialized the company admin menu mapping tables for each company. • Profile factory successfully loads the missing

	companies and their respective locations' information from database (see Figure 6.9).

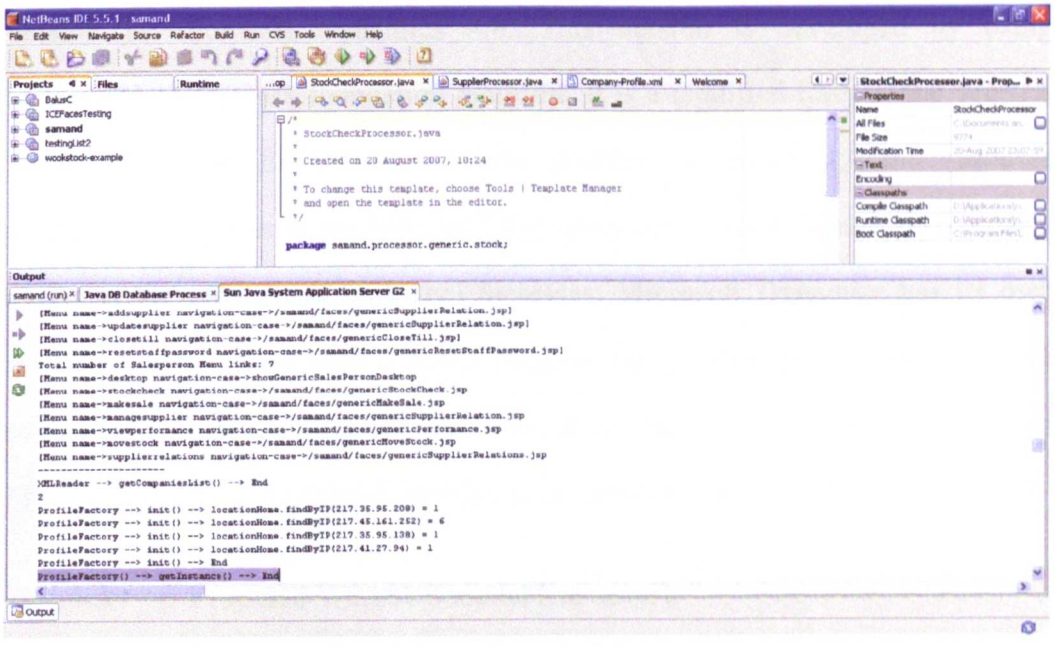


Figure 6.9: Profile factory initialization confirmation

6.2.4.1 Zoho CRM framework

ZohoCRM is using the existing Web 2.0 framework and in the Web 2.0 framework all the user interfaces are directly bounded to the business logic. Due to this architectural deficiency, all companies that run their e-business on ZohoCRM require their own dedicated set of views with the names of required services embedded inside them. The increase in views (interfaces) increases the memory usage on the hosting server and ultimately it causes memory shortage at the hosting server, increasing the response time and reducing the overall

performance etc. The result of the ZohoCRM scalability validation is shown in section 6.2.4.4.

6.2.4.2 Microsoft Dynamics

Microsoft Dynamics is available in different versions and currently the most used version is Microsoft Dynamics 4.0. This version is developed by using Microsoft's own work foundation platform. It is a modified form of the Web2.0 framework. This foundation platform provides a scalability and resource management options in the Microsoft dynamics but it still increases the memory usage as the view (user interface) increases. Microsoft Dynamics has its own resource management feature which reduces the memory usage from the other source as the memory usage increases from the views. The result of the Microsoft Dynamic scalability validation is shown in section 6.2.4.4.

6.2.4.3 Proposed W2ASVB model framework

The W2ASVB framework replaces the supporting functions with the service adapters. Service adapters are light weight and contain an implementation of the workflows. Following a user request service, adapters are connected to the core platform to execute the workflow modelled inside them. All workflows (both generic and customized) are modelled as service adapters. This feature provides an option in the W2ASVB framework to support a large of number of views (user interfaces) without consuming lot of memory. The W2ASVB framework provides a shared interface and performs user requests on the basis of profile configuration.

6.2.4.4 Results

The results of the scalability validation are given below.

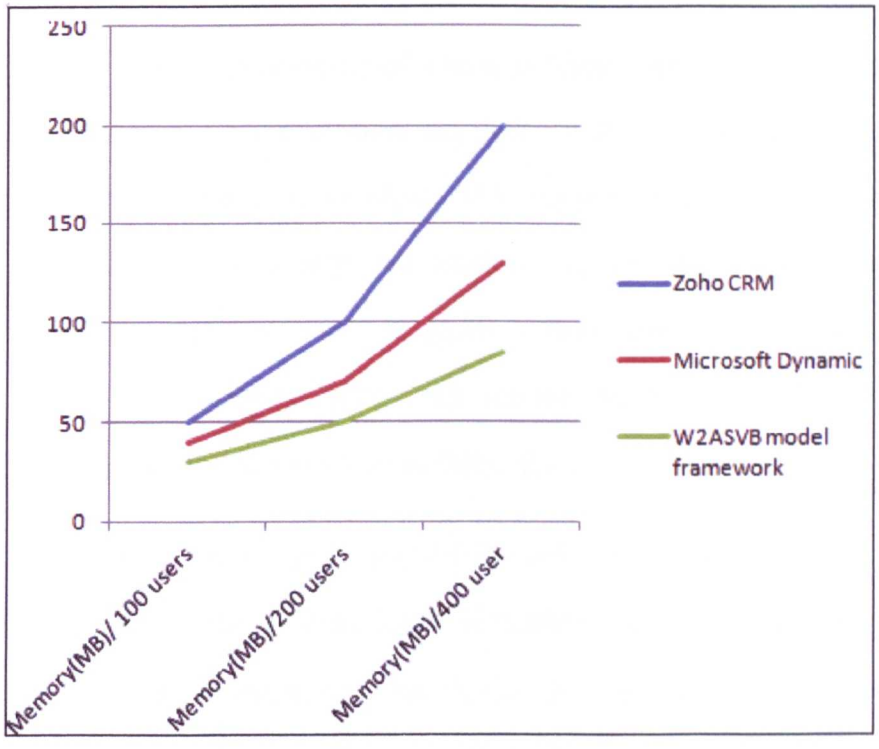


Figure 6.10: Results of the scalability validation

Figure 6.10 shows the results of the scalability validation. The W2ASVB framework memory usage/user decreases gradually by increasing the number of users. The reason for less memory usage at larger number of users is that W2ASVB is using the service adapters rather than normal service functions. These adapters release the memory that was occupied for performing the tasks after execution.

6.3 Advantages of the proposed solution

- The major contributions of the proposed system are the introduction of two brand new concepts; a service called service adapters and a request management mechanism called a request broker. As opposed to traditional web services, service adapters are light weight services and have shown better performance at run-time. This performance gain is a result of elimination of discovery and binding supporting functions from the SOAW2 model framework. In addition, they have proven their strategic importance in scenarios where new services are required to be built from existing web services without violating the SOA principles.

- The system enjoys good scalability and performance because of its enhanced plug-n-play capabilities, if implemented according to chapter 5. The use of plug-n-play hardware devices is very common, but in the field of software engineering this concept is quite new. Having software that contains such a feature is still rare. Internal architectural variations of different modules and absence of well-defined interfaces are the two biggest reasons for this lack of achievement until now. W2ASVB has overcome both of these deficiencies in a simple yet elegant and cost-effective way. The data dictionary modules in W2ASVB contain a list of modules and their functionalities information that is in use by different companies. Request brokers efficiently load them upon request at run-time. At any stage, if new modules are required to be plugged-in in the system, a service provider only requires compilation and entry of new module names in the data dictionary and the rest is handled automatically

by the intelligent brokers of the system. This new technique proved to be better in handling complex system evolution problems as faced by current retail systems. As a benefit of this technique, any number of new modules can now be unplugged from the system and easily be replaced with new ones. Conventional Web 2.0 systems do not exhibit such capabilities.

- One of the main aims of this project was to develop a generic set of user interfaces and business workflows. The reason behind this is to facilitate small retailers with ready-made solution for achieving instant e-business transformation. Traditional web systems do not provide such generic facilities as they are built on specific requirements of any particular company. Since customization of any existing system to suit any other companies' business process requires recompilation of the complete system code and therefore such a system becomes unaffordable to both solution provider and client companies. This situation becomes more cost-centric when a system is a desktop system and requires additional hardware installation.

- W2ASVB is equipped with basic essential features such as customized user interfaces, customised workflow mechanisms, profile management etc. that a small retailer needs; that is to put a first step on the e-business ladder at an affordable cost. The system does not require any special set up on the client side (such as hardware installation etc) and is accessible from any point in the world where Internet access is available. Small retailers who are interested in automation of their manual POS business

procedures need at minimum a standard desktop PC, a printer and a standard Internet connection to start their e-business.

- W2ASVB compared to other traditional Web 2.0 based systems offers a complete flexibility in its architectural design. Conventional web systems always suffer from the risk of undesirable architectural modification during upgrade and maintenance activities. W2ASVB architecture has been designed in such a way that any system evolution due to customization of either business logic or user interface does not pose any risk to core system design. In addition, and due to the system's central administrative nature, W2ASVB also offers central customization of business logic and user interfaces on a request from the end users. Thus, no visits of any personnel on client locations are required. This is a huge benefit of the proposed system and its implemented proposed model framework as it helps service provider in reducing maintenance and administration cost.

6.4 Shortcomings of proposed solution

The proposed solution, although it provides many advantages, also has some shortfalls. This section presents the disadvantages of the proposed solution and as the reader goes through these shortcomings, it will be noticed that they are very minor as compared to the advantages (i.e. as explained above) and only require small improvements.

- There is no mechanism for the management of request broker pool size in the proposed framework; at present the system demands a declaration of

constant request broker's pool size at a system initialization time. However, as the system has shared features mechanism, the responsibility for managing this pool size during busy times is on the system administrator's shoulders. Application server logs can help the system administrator in making this decision, but it requires extensive real-time human calculations. Errors in this calculation could have very serious impact on system performance.

- Although all the user requests pending in the waiting area are assured that they will be allocated request brokers as soon as they become available, no assurance is given on whether this allocation is made on the first come first served basis. As a matter of fact, prioritization of incoming request is an issue of application server; therefore, it should be researched at application server level rather than system level. Research at application server level requires extensive experience, a longer research period and a well-documented knowledge base from a server vendor. Unfortunately server vendors do not share their internal server architecture publicly; therefore this request prioritization issue can be classified as a shortcoming of the proposed system, but its solution is outside the scope of this project.

7. Conclusions and Future Recommendations

7.1 Contribution of the work

The proposed work seeks to make a contribution to knowledge and understanding in the field of Web 2.0 technologies and business process models, through the development of a particular area of theory and related application. Technically, it is intended to propose a new framework associated with specific optimisation methods in order to provide the customizable platform after reviewing both Web 2.0 technology (e.g. information sharing, interoperability, user- centre design and collaboration) and business process models (e.g. strategies, and operational processes). In practice it is intended to develop a 'framework' based on an efficient and secure model to facilitate small retail businesses in achieving e-business transformation. The framework is capable of being customized and scalable according to user requirements and reducing the overall installation cost.

7.1.1 Request handling

In this research by taking into consideration the shortcoming of request management components in SOAW2 and a need for an effective and intelligent request brokerage mechanism to handle complex on-demand sharing and customization problems, an enhanced request broker architecture is integrated between the user interface and the resource container. This proposed request broker architecture followed the standards mentioned in research by (Alur, Curpi, & Malks, 2003) but overcame limitations such as memory leaking and resource management etc. These proposed request brokers are responsible for action and view management inside the W2ASVB model framework. With the help of this proposed architecture, the W2ASVB framework provides an on-demand request routing between user interfaces and services of the core platform.

7.1.2 Service adapter

In this research, if workflows get modelled as web services, it would be a violation of loose-coupling and no embedded calls principle of SOA. To overcome these limitations and to avoid any conflict with SOA principles, workflows are modelled as service adapters. Following a user request for processing the adapters get connected with the core platform to execute workflow modelled inside them. All the workflows including both generic and customized are modelled as services adapters.

7.2 Evaluation of research question

The following conclusions are derived, related to the initial research question prior to the main investigation.

7.2.1 User request management modelling

In this thesis, it is shown that user requests for data processing can be divided into two parts namely data and action part. The data part contains the *data* that requires processing, whereas the action part describes the required *Operation* on the given *data* and the request brokers are semi-autonomous objects which are completely equipped with all the capabilities to handle action and view management for user requests.

7.2.2 Workflow modelling

The workflow management mechanism is introduced in W2ASVB to separate and segregate operations into two categories, namely generic and customized workflows. Also, the core business logic (i.e. data persistence logic) is embedded

in these generic and customized workflows as it helps to make a loosely coupled business logic and user interface.

7.3 Research contributions

- In order to overcome the problem of service and user interface sharing, it was found through the highlighted literature review that most of the request mediation architectures are very useful in handling such complex scenarios. By taking into consideration the shortcoming of request management components in SOAW2, and the need of effective and intelligent request brokerage mechanisms to handle complex on-demand sharing and customization issues, new request broker architecture is integrated between the user interface and the resource container. The proposed framework consists of a new request broker architecture concept. This request broker concept improves request handling process, which is responsible for an action and view management inside W2ASVB model framework. It provides on-demand request routing between user interfaces and services of core platform. The knowledge layer of the model framework is shared between service provider and users (i.e. employees of retail companies). This knowledge layer is the essential part of the Web 2.0 model framework and it is there to represent architecture of participation. This means that new knowledge in the model will be modified with the passage of the time.

- The research has also analysed the functional and non-functional dependencies in the shared systems. These dependencies appear as an

effect of construction of new services from existing functionalities. It is marked that in some situations due to the irreducible complexity of the scenario, compliance with strict SOA principles becomes difficult. The research answered this issue with a practical implementation of service adapters. The service adapter is a new concept of "light weight" services and contains the implementation of workflows. In SOA tradition, services are relatively large, shared, intrinsically loosely-coupled units of functionalities, and have no embedded calls to each other as W2ASVB model framework workflows are modelled as service adapters.

- This research also analysed the mechanism to control the generic workflow for each client as per its own requirements. In order to achieve this task, the proposed framework introduces the concept of profile management in XML format. It is one of the most powerful and self-descriptive languages in representing complex SOA-related data structures for brokerage. Furthermore, it can also be concluded that due to the profile management an elegant customization capability introduced into the proposed model framework, it reduces the development efforts of the solution provider company. This indeed results in the reduction of administration and development costs. Therefore, the model makes the system affordability two-dimensional i.e. it is not only affordable to small retail companies but is also affordable to the solution provider company.

- Another important outcome of this research is that the proposed framework is built independently of any existing e-business architecture.

The W2ASVB framework is a business domain dependent system, but the model framework itself is independent of any business domain. Therefore, it can be concluded that the proposed W2ASVB model framework is applicable to any relevant business domain to produce commercial scaled, shared, robust, flexible and affordable e-business solutions. It also reduces the operational cost dramatically due to its reusable nature and customizable business logic as already mentioned in the validation chapter.

- One of the main features of the proposed system is its on-demand customization capabilities. Contrary to traditional web systems, this customization facility is not only limited to the customization of business logic, but it can also help retailers in customizing the look of their business data i.e. customization of user interfaces.

7.4 Future recommendations

As an important activity of any research study, this section presents the issues that were highlighted as future research elements and require further investigation. Effective solution of these issues will contribute towards the improvement of the proposed model framework and will provide enhancements.

One future research area is the need of a request management algorithm to control the request brokers' pool size at run-time. Taking into consideration the highly operational and business-centric nature of the system, there is a need for serious investigation of this issue. Existing data mining algorithms would be useful, but a careful review is required to develop an improved version that not

only performs the run-time statistical calculation of incoming requests, but also uses its own knowledge base to decide the pool size. This new demand-and-supply algorithm must ensure that this pool size is set according to the relevant parameters such as available server resources and physical health of the system. It must also ensure that the system makes the best use of its logical resources (i.e. request brokers) to honour the maximum number of requests in a given time frame. Further research in this area can be done by implementing this framework by using the standard web service orchestration. In this case, the framework will inherit all the existing web service features, and these framework services can be used as external services.

Another possible future research area is the request prioritization issue. This issue was also amongst the shortcomings presented in the validation chapter. Further research is needed to come up with an effective algorithm which will ensure that the requests that are waiting longer get first priority. This is not a software level issue, it is rather an application hosting server level issue, and an effort can be made to resolve this efficiently.

8. References

B. Martinez, "Exploiting Social Tagging in a Web 2.0 Recommender System", *IEEE Journal on Internet Computer Society*, vol. 14, pages 23-30, 2010.

B. Choi, T. S. Raghu, A. Vinze, and K.J. Dooley, "Process Model for e-Business standards development: A Case of ebXML Standards", *IEEE transactions on engineering management*, vol. 56, pages 448-467, 2009.

B. Dubney, J. Lehr, B. Willis, and L. Mattingly, "Mastering JavaServer Faces", *1st Edition. Wiley Publishing Inc*, ISBN 0471462071, 2004.

B.C.C. Tan, S.L. Pan, R. Hackney, "The Strategic Implications of Web Technologies: A Process Model of How Web Technologies Enhance Organizational Performance", *IEEE Transactions on Engineering Management*, vol. 57, pages 181-197, 2010.

C. Leon and H. Enrique, "Virtual Service Grids: Integrating IT with Business Processes" , *IT Professional,* vol. 11, pages 7-11, 2009.

C. Schroth, "Web 2.0 versus SOA: Converging Concepts Enabling Seamless Cross-Organizational Collaboration", *4th IEEE International Conference on Enterprise Computing*, pages 47-54, 2007.

C. Pautasso, O. Zimmermann and F. Leymann, "RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision", *International word wide web conference*, April 21–25, 2008, Beijing, China, ACM 978-1-60558-085-2/08/04.

D. Breitgand, R. Cohen, A. Nahir, and D. Raz, "On Cost-Aware Monitoring for Self-Adaptive Load Sharing", *IEEE journal on selected areas in communications*, vol. 28, pages 70-83, 2010.

D. Guinard, V. Trifa and S. Karnouskos, "Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, andOn-Demand Provisioning of Web Services", *IEEE transactions on services computing journal*, vol. 3, pages 223-235, 2010.

D. Hinchcliffe, "Web 2.0 Continues Its Move To The Workplace".Reterived 12 03, 2011 from http://www.zdnet.com/blog/hinchcliffe/significant-workplace-inroads-for-enterprise-2-0/150.

D. Khazanchi and B. Munkvold. "On the Rhetoric and Relevance of IS Research Paradigms: A Conceptual Framework and Some Propositions". *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*, January 06 – 09, 2003, Big Island, Hawaii, pp. 252b.

E. Zeiris and M. Ziema, "SOA based e-business systems design", *Proceedings of the 2010 International Conference of e-business (ICE-B)*, pages 1-8, 2010.

F. Barbon, P. Traverso, M. Pistore, and M. Trainotti, "Run-Time Monitoring of Instances and Classes of Web Service Compositions", Proceedings of IEEE 6th International Conference on web Services, pages 63-71, 2006.

F. Sinton, "Enhancing an Application Server to Support Available Components", *IEEE Journal on Software Engineering*, vol.34, pages 531-545, 2008.

G. Carraro, "Software as a Service (SaaS): An Enterprise Perspective,"

Globus, "Globus Service Grid", Retrieved at 13 November 2011 from http://www.globus.org/ogsa.

H. Chen, "Trends & Controversies", *IEEE Journal on Intelligent System*, vol. 25, pages 68-83, 2010.

H. Zhao and H. Tong, "A Dynamic Service Composition Model Based on Constraints", *Sixth International Conference on Grid and Cooperative Computing (GCC 2007)* , pages 659-662, 2007.

H. Yuan, S.W. Choi, and S.D. Kim, "A Practical Monitoring Framework for ESB-Based Services", *Proceeding of IEEE Congress Services Part II (SERVICES-2 '08)*, pages 49-56, 2008.

H.-M. Chen, "SOA, Enterprise Architecture, and Business-IT Alignment: An Integrated Framework", *Proceeding of software Eng. research and practice conference*, pages 566-573, 2007

H. XiaoQin, H. LinPeng, C. Lin, and L. Minglu, "Design and Implementation of an Agent-Based Web Services Platform for Electronic Commerce. Services Computing", *IEEE International Conference on Services Computing (SCC'04)*, pages 643-646, 2004.

J. Dorn, A. Rainer, and P. Hrastnik, "Toward Semantic Composition of Web Services with MOVE", *8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services*, page 67, 2006.

J. Asensio, J. Vergara, , and J. Berrocal, "Experiences with SNMP based integrated management of CORBA-based elecronic commerce application", *Sixth IFIP/IEEE International Symposium on Integrated Network Management*, Page(s): 517 - 530 ,1999.

J. Zhao, W.V. Huang, and Z. Zhu, "An Empirical Study of E-Business Implementation Process in China", *IEEE journal on engineering management transactions*, vol. 55, pages 134-147, 2008.

J. Xu, D. Zhang, L. Liu and X. Li, "Dynamic Authentication for Cross-Realm SOA-Based Business Processes", *IEEE transactions on services computing*, vol. 5, pages 20-22, 2012.

J. Simmonds, Y. Gan, M. Chechik, S. Nejati, B. O'Farrell, E. Litani and J. Waterhouse, "Runtime Monitoring of Web Service Conversations", *IEEE Trans. Services Computing*, vol. 2, pages 223-244, 2009.

J. Koskinen. "Software maintenance cost estimation and modernization support". Retrieved 08 02, 2011, from http://www.citeulike.org/user/mattbiehl/article/7530373.

L. Hitachi, "Proposal of Application Architecture in Electronic Commerce Service between Companies". *International conference on advance issues of e-commerce and web-based information systems* , page 46-49, 1999.

L. Baresi and S. Guinea, "Towards Dynamic Monitoring of WSBPEL Processes", proceeding of 3rd International Conference on Service-Oriented Computing (ICSOC '05), pages 269-282, 2005.

L.D. Xu, "Enter Systems: State of the Art and Future Trends", *IEEE Transcations on industrial informations*, Vol. 7, 2011.

L.M.S. de Souza, P. Spiess, D. Guinard, M. Ko¨hler, S. Karnouskos, and D. Savio, "SOCRADES: A Web Service Based Shop Floor Integration Infrastructure", *Proc. Internet of Things Conf. (IoT '08)*, pages 50-67, 2008.

M. Bichler and M. Kaukal, "Design and Implementation of a Brokerage Service for Electronic Procurement". *10th International Workshop on Database & Expert Systems Applications* , pages 618-622, 1999.

M..J. Carey, "Service Oriented Architecture (SOA) What?", *IEEE journal on Internet Computing*, vol. 41 , pages 92-94, 2008.

M.A. Davidson, E. Yoran, "Enterprise Security for Web 2.0", *IEEE Journal on Computers*, Vol. 40, pages 117-119, 2007.

Microsoft Corp., Retrieved at 12 September 2011 from http://msdn.microsoft.com/en-us/library/ aa905332.aspx.

MerchantOS. (2011). Introduction to MerchantOS. Retrieved 7 12, 2011, from MerchantOS: http://www.merchantos.com/try-it/2/

M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, and P. Traverso, "Planning and Monitoring web Service Composition", *Artificial Intelligence Methodology, Systems, and Applications*, vol. 3192, pages 106-115, 2004.

Milpied and Dubois, "Arrhythmia Discrimination in Implantable Cardioverter Defibrillators Using Support Vector Machines Applied to a New Representation

of Electrograms", *IEEE Journal on Biomedical Engineering*, vol. 58, pages 1797-1803, 2011.

M. Kano, A. Koide, T.K. Liu, and B. Ramachandran, "Analysis and simulation of business solutions in a service-oriented architecture", *IBM System Journal*, vol. 44, pages 669–690, 2005.

M. Shafiq, Y. Ding and D. Fensel, "Bridging Multi Agent Systems and Web Services: towards interoperability between Software Agents and Semantic Web Services", *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*, pages 85-96, 2006.

N. Bieberstein, S. Bose, L. Walker, and A. Lynch, "Impact of service oriented architecture on enterprise systems, organizational structures, and individuals", IBM System journal, vol. 44, pages 691–708, 2005.

O. Nasraoui and M. Soliman, "A Web Usage Mining Framework for Mining Evolving User Profiles in Dynamic Web Sites", *IEEE Journal Knowledge and Data Engineering*, vol. 20, pages 202-215, 2008.

O. Mahmood, "Developing Web 2.0 Applications for Semantic Web of Trust", *International Conference on Information Technology (ITNG'07)*, pages 819-824, 2007.

P. Felber, P. Narasimhan, "Experiences, strategies, and challenges in building fault-tolerant CORBA systems", *IEEE journal on computers Transactions*, vol. 53, pages 497-511, 2004.

PHPPointOfSale. (2011). Peoria Plumbing Supply. Retrieved 7 12, 2011, from PHP Point of Sale: http://demo.phppointofsale.com/index.php

P. Pankaj, "An analysis and exploration of the construct of information systems agility", *Ph.D. dissertation, Southern Illinois Univ. Carbondale, Carbondale, IL*, 2004.

P. Xiong, Y. Fan, and M. Zhou, "QoS-aware web service conguration", *IEEE Transaction System*, vol. 38, pages 888–895, 2008.

P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L.M.S.d. Souza, and V. Trifa, "SOA-Based Integration of the Internet of Things in Enterprise Services", *Proceeding of IEEE International Conference on Web Services (ICWS '09)*, pages 968-975, 2009.

Q. Wang, J. Shao, F. Deng, Y. Liu, M. Li, J. Han, and H. Mei, "An Online Monitoring Approach for Web Service Requirements", *IEEE Transaction Services Computing*, vol. 2, pages 338-351, 2009.

R. Battle, "Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST)", *Journal of Web semantics: Science, Services and Agents on the World Wide Web*, vol. 6, pages 61-69, 2008.

R. Howard and L. Kerschberg, "A Knowledge-based Framework for Dynamic Semantic Web Services Brokering and Management. Database and Expert Systems Applications", *15th International Workshop on database and expert systems applications* , pages 174-178, 2004.

R. Pressman, "Software Engineering: Practitioner's Approach". McGraw Hill Inc, ISBN 0071267824, 2005.

S. Aughton, "Controlling the Business Growth Speed from Financial Angle of View: Using Cash Flow System Dyanmics Model", *IEEE conference on First International Workshop on Database Technology and Applications* , page 610-613, 2009.

S. Kuk, I. Oh, H. Kim, J.K. Lee, and S.W. Park,"An e-Engineering Framework Based on Service-Oriented Architecture and Agent Technologies", *Proceedings of the 2007 11th International Conference on Computer Supported Cooperative Work in Design*, pages 429-434, 2007.

S. Kuk, I. Oh, H. Kim, J.K. Lee, and S.W. Park. "Service-Oriented Architecture Based e-Engineering Framework to Support Collaborative Design", *IEEE International Conference on Services Computing (SCC 2007)*, pages 340-347, 2007.

S. Murugesan, "Understanding Web 2.0", IT Pro, vol. 9, pages 34-41, 2007.

S. Shenoy and N. Mallya, "Integrating Struts, Tiles, and JavaServer Faces", Retrieved 03 27, 2007, from IBM Research Center: http://www-128.ibm.com/developerworks/library/j-integrate.

S. Karnouskos, O. Baecker, L.M.S.d. Souza, and P. Spiess, "Integration of SOA-Ready Networked Embedded Devices in Enterprise Systems via a Cross-Layered Web Service Infrastructure", *Proceeding of IEEE Conference of Emerging Technologies & Factory Automation (ETFA)*, pages 293-300, 2007.

T. Shan, "Taxonomy of Java Web Application Frameworks", *IEEE International Conference on e-Business Engineering*, pages 378-385, 2006.

T. Erl, "Service-Oriented Architecture: Concepts, Technology, and Design", Prentice Hall PTR, ISBN 0131858580, 2005.

T.C. Shan and W.W. Hua, "Service-Oriented Solution Framework for Internet Banking", *International Journal of Web Services Research*, vol. 3, pages 29-48, 2006.

V. Oosterhout, E. Waarts, and J.V. Hillegersberg, "Change factors requiring agility and implications for IT", *European Journal of Information System*, vol. 15, pages 132–145, 2006.

V. Sambamurthy, A. Bharadwaj, and V. Grover, "Shaping agility through digital options: Reconceptualizing the role of information technology in contemporary rms", *MIS Quart.*, vol. 27, pages 237–263, 2003.

V.A.F. Almeida, "Internet Workloads: Measurement, Characterization, and Modeling" , *IEEE journal on Internet computing* , vol. 15, page 15-18, 2011.

W. Freiler, "Interactive Visual Analysis of Set-Typed Data", *IEEE journal on Visualization and Computer Graphics*, vol. 14, pages 1340- 1347, 2008.

W. J. Clarke, L. C. Alves, "IBM System z10 design for RAS", *IBM Journal of Research and Development*, vol. 53, pages 11-22, 2010.

W. Marin, "Remote Programming of Network Robots Within the UJI Industrial Robotics Telelaboratory: FPGA Vision and SNRP Network Protocol", *IEEE Journal on Industrial Electronics*, vol. 56, pages 4806-4816, 2009.

W. Omar, A. Abbas, and T. Bendiab, "SOAW2 for Managing the Web 2.0 Framework", *IEEE Journal on Computer Society*, Vol. 9, pages. 30-35, 2007.

WC3. (2004). "Web Services Specifications", Retrieved 7 28, 2010, from Web Services Architecture: http://www.w3.org/TR/ws-arch/.

X. Li, Y. Fan, Q. Z. Sheng, Z. Maamar, and H. Zhu, "A petri net approach to analyzing behavioral compatibility and similarity of web services", IEEE Transaction System, vol. 41, pages 510–521, 2011.

Y. Liu, I. Gorton, and L. Zhu, "Performance Prediction of Service-Oriented Applications Based on an Enterprise Service Bus", *Proceeding of 31st annual Internation computer software and applications Conference (COMPSAC'07)*, pages 327-334, 2007.

Y. Dang, Y. Zhang, H. Chen, "A Lexicon-Enhanced Method for Sentiment Classification: An Experiment on Online Product Reviews", *IEEE Journal on Intelligent System*, vol. 25, pages 46-53, 2010.

Y. Juan and W. Hongxia, "Study on E-business Logistics System Based On SOA", *Computer Science and Information Technology (ICCSIT) journal*, vol. 3, pagess 368 - 372, 2010.

ZohoCRM , "Zoho CRM Introduction" Retrieved 7 12, 2010, from Zoho CRM website: http://crm.zoho.com/crm/ShowHomePage.do.

Z. Sen, H. Shuangxi and F. Yushun, "Service-Oriented Enterprise Network Performance Analysis", *tsinghua science and technology journal*, vol 14, pages 492 -503, 2009.

9. Abbreviations

ADSS	Autonomous Decentralized Service System
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CBPI	Company Business Process Infrastructure
CORBA	Common Object Request Broker Architecture
EJB	Enterprise Java Beans
EIS	Enterprise Information System
GUI	Graphical User Interface
HCI	Human Computer Interaction
ISO	International Standard Organization
IEC	International Electro technical Commission
IEEE	Institute of Electrical & Electronics Engineering
IP	Internet Protocol
KDSWS	Knowledge-based Dynamic Semantic Web Services
NS	Networked Services
OMG	Object Management Group
OOAD	Object Oriented Analysis and Design
OLTP	Online Transaction Platform

POS	Point of Sale
RMI	Remote Method Invocation
RTSOA	Real-time Service-oriented Architecture
RUP	Rational Unified Process
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TINA	Telecommunications Information Networking Architecture
User	A user interacting with a system
UI	User Interface
UML	Unified Modelling Language
WS	Web Services
WSDL	Web Services Definition Language
W2ASVB	Web 2.0 Architecture for Service and View Brokerage
XML	Extensible Mark-up Language

10 Appendix

10.1 Source Code

10.1.1 SystemManager.java

```
/*
 * SystemManager.java
 *
 * Created on 28 June 2012, 23:50
 */
package w2asvb;

import com.sun.rave.web.ui.appbase.AbstractApplicationBean;
import java.util.Vector;
import javax.faces.FacesException;
import samand.broker.WebRequestBroker;
import samand.exception.AppException;
import samand.exception.ProfileFactoryException;
import samand.profile.ProfileFactory;
import samand.util.Constants;
import samand.util.DEBUG;

/**
 * <p>Application scope data bean for your application.  Create properties
 *  here to represent cached data that should be made available to all users
 *  and pages in the application.</p>
 *
 * <p>An instance of this class will be created for you automatically,
 * the first time your application evaluates a value binding expression
 * or method binding expression that references a managed bean using
 * this class.</p>
 */
public class SystemManager extends AbstractApplicationBean {
    // <editor-fold defaultstate="collapsed" desc="Managed Component Definition">
    private int __placeholder;

    /**
     * <p>Automatically managed component initialization.  <strong>WARNING:</strong>
     * This method is automatically generated, so any user-specified code inserted
     * here is subject to being replaced.</p>
     */
    private void _init() throws Exception {
    }
    // </editor-fold>

    /**
     * <p>Construct a new application data bean instance.</p>
     */
    public SystemManager() {
    }

    public void init() {
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "init()", "Start");
        // Perform initializations inherited from our superclass
        super.init();
        // Perform application initialization that must complete
        // *before* managed components are initialized

        // <editor-fold defaultstate="collapsed" desc="Managed Component
Initialization">
        // Initialize automatically managed components
        // *Note* - this logic should NOT be modified
        try {
            _init();
        } catch (Exception e) {
            log("ApplicationBean1 Initialization Failure", e);
            throw e instanceof FacesException ? (FacesException) e: new
FacesException(e);
        }
```

```
        // </editor-fold>
        //1. Initialize EJB Platform
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "init()", "Initializing EJB
Platform");
        samand.services.ResourceRegisterar.registerEJBS();
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "init()", "EJB Platform
Initialization complete.");

        //2. Initialize profile factory
        try {
            DEBUG.println(DEBUG.Debug_Level, "SystemManager", "init()", "Initializing
ProfileFactory.");
            this.profileFactory = ProfileFactory.getInstance();
            DEBUG.println(DEBUG.Debug_Level, "SystemManager", "init()",
"ProfileFactory initialization complete.");

        } catch (ProfileFactoryException e) {
            DEBUG.println(DEBUG.Debug_Level, "SystemManager", "init()",
"ProfileFactory initialization Failed with exception::" + e);
            //setting up PageAlert bean to be displayed on logon page
            PageAlertBean pageAlertBean = (PageAlertBean) getBean("PageAlertBean");
            pageAlertBean.setType("error");
            pageAlertBean.setTitle("System Error");
            pageAlertBean.setSummary("Internal System Error Occured. Please inform at
it@rankhour.com");
            pageAlertBean.setVisible(true);
        }
        //3. Initialize request brokers pool
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "init()", "About to call
initRequestBrokersPool().");
        initRequestBrokersPool();
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "init()", "End");
    }//end of init()

    public void destroy() {
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "destroy()", "Start");

        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "destroy()", "About to
destory EJB Platform Resources.");
        samand.services.ResourceRegisterar.unregisterResource();
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "destroy()", "EJB Platform
Resources destruction complete.");

        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "destroy()", "End");
    }

    public String getLocaleCharacterEncoding() {
        return super.getLocaleCharacterEncoding();
    }

    private ProfileFactory profileFactory;
    private Vector requestBrokersPool;

    /*
    public Vector getRequestBrokersPool() {
        return requestBrokersPool;
    }
    */
    public void setRequestBrokersPool(Vector requestBrokersPool) {
        this.requestBrokersPool = requestBrokersPool;
    }

    ////////HELPER METHODS
    public samand.profile.Location getLocation(String ipAddress){
        return(profileFactory.getLocation(ipAddress));
    }

    private void initRequestBrokersPool(){
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "initRequestBrokersPool()",
"Start");
        WebRequestBroker webReqBroker;
```

```
        int poolSize = Integer.parseInt(Constants.REQBROKER_POOL_SIZE);
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "initRequestBrokersPool()",
"POOL SIZE set to->" + poolSize);

        Vector list = null;
        list = new Vector();

        for(int i=0; i<poolSize; i++){
            webReqBroker = new WebRequestBroker();
            list.add(webReqBroker);
            DEBUG.println(DEBUG.Debug_Level, "SystemManager",
"initRequestBrokersPool()", "WebRequestBroker [" + webReqBroker + "] added in pool at
position->" +i );
        }

        this.requestBrokersPool = list;

        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "initRequestBrokersPool()",
"End");
    }

    //This method will be called by JSFActionDispatcher to get free WebRequestBroker

    public synchronized WebRequestBroker getWebRequestBroker() {
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "getWebRequestBroker()",
"Start");

        boolean freeBrokerFound = false;
        WebRequestBroker broker = null;

        while(!freeBrokerFound){
            for(int i=0;i<this.requestBrokersPool.size(); i++){
                broker = (WebRequestBroker)requestBrokersPool.get(i);
                if(!broker.isBusy()){
                    freeBrokerFound = true;
                    broker.setBusy(true);
                    break;
                }
            }//end of inner for
        }//end of while
        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "getWebRequestBroker()",
"Broker [" + broker + "] is being allocated to honour this request.");

        DEBUG.println(DEBUG.Debug_Level, "SystemManager", "getWebRequestBroker()",
"End");
        return broker;
    }
}
```

10.1.2 ProfileFactory.java

```
/*
 * ProfileFactory.java
 *
 * Created on 29 June 2012, 19:13
 *
 */

package w2asvb.profile;

import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Vector;
import samand.exception.ProfileFactoryException;
import samand.services.EJBReferences;
import samand.services.ResourceRegisterar;
import samand.util.Constants;
import samand.util.DEBUG;
import samand.util.XMLReader;

/**
 */
public class ProfileFactory {

    public static ProfileFactory ourInstance = null;
    private HashMap<String, Location> hmLocationProfiles;


    /** Creates a new instance of ProfileFactory */
    public ProfileFactory() throws Exception{
        init();
    }

    public void init() throws Exception{
        DEBUG.println(DEBUG.Debug_Level, "ProfileFactory", "init()", "Start");

        //1. initiate location profile hashmap (IP, Location)
        hmLocationProfiles = new HashMap<String, Location>();

        //2. calling xmlreader to read file company profile file
        List<samand.profile.Company> companiesList =
XMLReader.getInstance().getCompaniesList(Constants.COMPANY_PROFILE_FILE_PATH);

        //3. Iterating List one by one filling missing informations in Location and
Company Object
        //and then creating location based hashmap
        samand.profile.Company companyObj; samand.profile.Location location;
        orms.company.CompanyRemoteHome companyHome; orms.location.LocationRemoteHome
locationHome;
        orms.company.Company companyRemote; orms.location.Location locationRemote;
        Vector locationVector;
        Collection tempCollection;

        System.out.println(companiesList.size());


        /*************************PLATFORM CODE*****************************/
        for(int i = 0; i < companiesList.size(); i++ ){
            companyObj = companiesList.get(i);

            try{
                //locking up company home reference

                companyHome =
(orms.company.CompanyRemoteHome)ResourceRegisterar.lookupResource(EJBReferences.COMPAN
Y_HOME_NAME);
```

```
                companyRemote =
companyHome.findByPrimaryKey(companyObj.getCompanyID());

                companyObj.setCompanyName(companyRemote.getCompName());

                //locking individual location objection inside each company
                //and retreving and storing values
                locationVector = companyObj.getLocations();
                for(int j = 0; j<locationVector.size(); j++){
                    location = (Location)locationVector.get(j);

                    try{
                        locationHome = (orms.location.LocationRemoteHome)
ResourceRegisterar.lookupResource(EJBReferences.LOC_HOME_NAME);
                        tempCollection =
locationHome.findByIp(location.getLocationIP());

                        DEBUG.println(DEBUG.Debug_Level, "ProfileFactory", "init()",
"locationHome.findByIP("+ location.getLocationIP() + ") = " + tempCollection.size());
                        locationRemote = (orms.location.Location)
tempCollection.toArray()[0];

                        //setting the location object values
                        location.setLocationID(locationRemote.getLocId());
                        location.setName(locationRemote.getLocName());
                        location.setAddr1(locationRemote.getAddress1());
                        location.setAddr2(locationRemote.getAddress2());
                        location.setCity(locationRemote.getCity());
                        location.setPostCode(locationRemote.getPostcode());
                        location.setTel(locationRemote.getTel());
                        location.setCompany(companyObj);

                        //adding location into hashtable<IPAddress, Location Object>
                        hmLocationProfiles.put(location.getLocationIP(), location);

                    } catch (Exception e){
                        DEBUG.println(DEBUG.Debug_Level, "ProfileFactory", "init()",
"Exception Occured[");
                        e.printStackTrace();
                        DEBUG.println(DEBUG.Debug_Level, "ProfileFactory", "init()",
"]");
                        throw new samand.exception.InitiationException("Profile
Factory: Company profile loading failed.");
                    }//end of inner try-catch
                }//end of inner for
            } catch (Exception e){
                DEBUG.println(DEBUG.Debug_Level, "ProfileFactory", "init()",
"Exception Occured[");
                e.printStackTrace();
                DEBUG.println(DEBUG.Debug_Level, "ProfileFactory", "init()", "]");
                throw new samand.exception.InitiationException("Profile Factory:
Company profile loading failed.");
            }//end of outer try-catch

        }//end of for loop
         /*********************************************************************/

        DEBUG.println(DEBUG.Debug_Level, "ProfileFactory", "init()", "End");
    }

    public void destroy() {
    }

    public static synchronized ProfileFactory getInstance() throws
ProfileFactoryException{
        DEBUG.println(DEBUG.Debug_Level, "ProfileFactory()", "getInstance()",
"Start");
        if(ourInstance == null)
            try{
                return new ProfileFactory();
```

```
} catch (Exception e){
                throw new ProfileFactoryException("Profile Factory Initiataion
Failed:[" + e + "]");

            } finally {
                DEBUG.println(DEBUG.Debug_Level, "ProfileFactory()", "getInstance()",
"End");
            }
        else {
            DEBUG.println(DEBUG.Debug_Level, "ProfileFactory()", "getInstance()",
"End");
            return ourInstance;
        }
    }

    public Location getLocation(String ipAddress){
        return(hmLocationProfiles.get(ipAddress));
    }

}
```

10.1.3 XMLReader.java

```
/*
 * XMLReader.java
 *
 * Created on 29 June 2012, 22:01
 *
 */

package w2asvb.util;

import java.util.*;
import java.io.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.xml.sax.SAXException;
import org.w3c.dom.*;
import samand.profile.ActionEntry;
import samand.profile.Company;
import samand.profile.Location;

/**
 *
 * @author Ather Mughal
 */
public class XMLReader {
    private static XMLReader ourInstance = null;

    public XMLReader() {
        init();
    }

    private void init(){
    }

    public List getCompaniesList(String filePath){
        DEBUG.println(DEBUG.Debug_Level, "XMLReader", "getCompaniesList()", "Start");

        List<Company> companiesList = new ArrayList<Company>();
        Company company;

        try{

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse( new File(filePath) );

            System.out.println("Root node of XML document is: " +
document.getDocumentElement().getNodeName());

            NodeList listOfCompanies = document.getElementsByTagName("Company");

            int totalProfiles = listOfCompanies.getLength();
            System.out.println("Total no of Company Profiles in XML file : " +
totalProfiles);


            //debug
            int numberOfIPAddresses = 0;

            for(int i=0; i<listOfCompanies.getLength(); i++){

                //selecting clients one by one
                Node companyNode = listOfCompanies.item(i);
                company = new Company();

                if(companyNode.getNodeType() == Node.ELEMENT_NODE){

                    //casting client Node into Element
                    Element companyElement = (Element)companyNode;

                    //Retreving company id and url attribute values
                    System.out.println("Company ID: " +
```

```
companyElement.getAttributes().getNamedItem("id").getNodeValue());
                    System.out.println("URL : " +
companyElement.getAttributes().getNamedItem("url").getNodeValue());


company.setCompanyID(companyElement.getAttributes().getNamedItem("id").getNodeValue())
;

company.setUrl(companyElement.getAttributes().getNamedItem("url").getNodeValue());

                    //------------Going to retrieve Locations
                    NodeList companyLocationsList =
companyElement.getElementsByTagName("Locations");
                    Element companyLocationElement =
(Element)companyLocationsList.item(0);


                    NodeList companyLocationNodes =
companyLocationElement.getElementsByTagName("Location");
                    System.out.println("Total number of registered Locations for this
Company : " + companyLocationNodes.getLength());

                    for(int j=0; j<companyLocationNodes.getLength(); j++){

                        Node locationNode = companyLocationNodes.item(j);
                        if(locationNode.getNodeType() == Node.ELEMENT_NODE){

                            System.out.println("Location IP-Address=" +
locationNode.getAttributes().getNamedItem("IP-Address").getNodeValue().trim() +
                                                " logopath=" +
locationNode.getAttributes().getNamedItem("logopath").getNodeValue().trim());
                            company.addLocation(new
Location(locationNode.getAttributes().getNamedItem("IP-
Address").getNodeValue().trim(),

locationNode.getAttributes().getNamedItem("logopath").getNodeValue().trim()));
                        }//end of inner-if
                    }//end of Locations



                    //------------Going to retrieve Action Processors  List
                    NodeList companyActionProcessorsList =
companyElement.getElementsByTagName("Action-Processors-Mapping");
                    Element companyActionProcessorElement =
(Element)companyActionProcessorsList.item(0);


                    NodeList companyActionProcessorNodes =
companyActionProcessorElement.getElementsByTagName("Action-Entry");
                    System.out.println("Total number of Action-Entry : " +
companyActionProcessorNodes.getLength());

                    for(int j=0; j<companyActionProcessorNodes.getLength(); j++){

                        Node actionEntryNode = companyActionProcessorNodes.item(j);
                        if(actionEntryNode.getNodeType() == Node.ELEMENT_NODE){

                            System.out.println("[Action-Entry source->" +
actionEntryNode.getAttributes().getNamedItem("source").getNodeValue().trim() +
                                                " dest->" +
actionEntryNode.getAttributes().getNamedItem("dest").getNodeValue().trim() +
                                                " actionProcessor->" +
actionEntryNode.getAttributes().getNamedItem("actionProcessor").getNodeValue().trim()
+
                                                " errorpage->" +
actionEntryNode.getAttributes().getNamedItem("errorpage").getNodeValue().trim() +
"]");


company.addActionMapEntry(actionEntryNode.getAttributes().getNamedItem("source").getNo
deValue().trim(),
```

```
                                                        new
ActionEntry(actionEntryNode.getAttributes().getNamedItem("source").getNodeValue().trim
(),

actionEntryNode.getAttributes().getNamedItem("dest").getNodeValue().trim(),

actionEntryNode.getAttributes().getNamedItem("actionProcessor").getNodeValue().trim(),

actionEntryNode.getAttributes().getNamedItem("errorpage").getNodeValue().trim()
                                                                              )
                                                    );
                        }//end of inner-if
                    }//end of Action-Processors-Mapping


                    //------------Going to retrieve Admin Menu Mapping
                    NodeList adminMenuList =
companyElement.getElementsByTagName("Admin-Menu-Mapping");
                    Element adminMenuElement = (Element)adminMenuList.item(0);


                    NodeList adminMenuNodes =
adminMenuElement.getElementsByTagName("Menu");
                    System.out.println("Total number of Admin Menu links: " +
adminMenuNodes.getLength());

                    for(int j=0; j<adminMenuNodes.getLength(); j++){

                        Node menuNode = adminMenuNodes.item(j);
                        if(menuNode.getNodeType() == Node.ELEMENT_NODE){

                            System.out.println("[Menu name->" +
menuNode.getAttributes().getNamedItem("name").getNodeValue().trim() +
                                                    " navigation-case->" +
menuNode.getAttributes().getNamedItem("navigation-case").getNodeValue().trim() + "]");


company.addAdminMenuItem(menuNode.getAttributes().getNamedItem("name").getNodeValue().
trim(),

menuNode.getAttributes().getNamedItem("navigation-case").getNodeValue().trim()
                                                    );

                        }//end of inner-if
                    }//end of Admin-Menu-Mapping


                    //------------Going to retrieve Sales Person Menu Mapping
                    NodeList salesPersonMenuList =
companyElement.getElementsByTagName("SalesPerson-Menu-Mapping");
                    Element salesPersonMenuElement =
(Element)salesPersonMenuList.item(0);


                    NodeList salesPersonMenuNodes =
salesPersonMenuElement.getElementsByTagName("Menu");
                    System.out.println("Total number of Salesperson Menu links: " +
salesPersonMenuNodes.getLength());

                    for(int j=0; j<salesPersonMenuNodes.getLength(); j++){

                        Node menuNode = salesPersonMenuNodes.item(j);
                        if(menuNode.getNodeType() == Node.ELEMENT_NODE){

                            System.out.println("[Menu name->" +
menuNode.getAttributes().getNamedItem("name").getNodeValue().trim() +
                                                    " navigation-case->" +
menuNode.getAttributes().getNamedItem("navigation-case").getNodeValue().trim());
```

```
company.addSalesPersonMenuItem(menuNode.getAttributes().getNamedItem("name").getNodeVa
lue().trim(),

menuNode.getAttributes().getNamedItem("navigation-case").getNodeValue().trim()
                                                                    );

                        }//end of inner-if
                    }//end of Admin-Menu-Mapping

                }//end of outer-if
                System.out.println("----------------------");

                //adding company to list
                companiesList.add(company);
            }//end of outer for

        } catch(NullPointerException e){
            throw new NullPointerException("Unable to locate given file:" + filePath
);
        } catch (SAXException sxe) {
            // Error generated during parsing
            Exception x = sxe;
            if (sxe.getException() != null)
                x = sxe.getException();
            x.printStackTrace();
        } catch (ParserConfigurationException pce) {
            // Parser with specified options can't be built
            pce.printStackTrace();

        } catch (IOException ioe) {
            // I/O error
            ioe.printStackTrace();
        }
        DEBUG.println(DEBUG.Debug_Level, "XMLReader", "getCompaniesList()", "End");
        return companiesList;
    }


    public static synchronized XMLReader getInstance(){
        DEBUG.println(DEBUG.Debug_Level, "XMLReader", "getInstance()", "Start");
        if(ourInstance == null){
            DEBUG.println(DEBUG.Debug_Level, "XMLReader", "getInstance()", "End");
            return new XMLReader();
        } else{
            DEBUG.println(DEBUG.Debug_Level, "XMLReader", "getInstance()", "End");
            return ourInstance;
        }
    }

}
```

10.1.4 Login.java

```
    public String login_btn_action() {
        DEBUG.println(DEBUG.Debug_Level, "login.java", "login_btn_action()", "Start");
        //0-retrieve user inputs
        String strUid = (String) this.getUserName().getText();
        String strPwd = (String) this.getUserPassword().getText();

        //reset the password for security
        this.getUserPassword().setPassword("");

        //getting role
        String strRole = (String)this.getDrp_login_as().getValue();

        //this statement will initiate system manager, if not already initaited
        SystemManager sysMgr = getSystemManager();

         //1- authenticate user details
        LoginProcessor loginProcessor = new LoginProcessor();
        UserSessionBean usb ;

        try {

            //2. calling to authenticate user credentials
            samand.profile.Employee emp = loginProcessor.authenticateEmployee(strUid,
strPwd);

            //Temporary code, making employee administator
            emp.setCompanyAdmin(true);

            //2.1 checking wheather person is allowed to be logged in as admin or not
            if(!emp.isCompanyAdmin() && strRole.equalsIgnoreCase("adminLogin")){
                PageAlertBean alertBean = (PageAlertBean)getBean("PageAlertBean");
                alertBean.setType("warning");
                alertBean.setTitle("User Access - Error:");
                alertBean.setSummary(Constants.INSUFFICIENT_PRIVILAGES_ERROR_MSG);
                alertBean.setVisible(true);

                // redisplaying page with page alert
                return null;
            }
            //3. calling to authenticate location access
            HttpServletRequest req=(HttpServletRequest)
            FacesContext.getCurrentInstance().getExternalContext().getRequest();
            String strAddr = req.getRemoteAddr();

            //temporaryly setting ip address to 217.35.95.208
            strAddr = "217.35.95.208";
            DEBUG.println(DEBUG.Debug_Level, "login.java", "login_btn_action()",
"Remote Location IP address is :" + strAddr);
            loginProcessor.authenticateLocation(strAddr);

            //4. On sucessfull completion, loading user details into UserSessionBean
            usb = getUserSessionBean();


            //5. Retreving and storing Location details of user from System Manager
            samand.profile.Location loc = sysMgr.getLocation(strAddr);

            usb.setLocation(loc);
            usb.setEmployee(emp);
            usb.setActionMenuMapping(loc.getCompany().getHmActionProcessorMapping());

            //6. Setting up menu mepping based on role
            if(strRole.equalsIgnoreCase("adminLogin")){
                DEBUG.println(DEBUG.Debug_Level, "login.java", "login_btn_action()",
"Setting up Admin Menu Mapping table in User Session Profile.");
                usb.setCurrentMenuMapping(loc.getCompany().getHmAdminMenuMapping());
            } else {
                DEBUG.println(DEBUG.Debug_Level, "login.java", "login_btn_action()",
"Setting up Sales Person Menu Mapping table in User Session Profile.");
```

```
usb.setCurrentMenuMapping(loc.getCompany().getHmSalesPersonMenuMapping());
            }

            //7. Navigating to target user desktop
            String temp = usb.getNavigationCase(Constants.USER_DESKTOP);
            return temp;
            //return "showTempPage";
        } catch (Exception e) {
            //setting up page alert to be displayed to user

            PageAlertBean alertBean = (PageAlertBean)getBean("PageAlertBean");
            alertBean.setType("error");

            if(e instanceof AuthenticationException){
                alertBean.setTitle("User Access - Error:");
                alertBean.setSummary(e.getMessage());
            } else if (e instanceof PlatformException){
                alertBean.setTitle("System Error:");
                alertBean.setSummary(e.getMessage());
            }

            alertBean.setVisible(true);
        }
        DEBUG.println(DEBUG.Debug_Level, "login.java", "login_btn_action()", "Start");

        return null;
    }//end of login button()
```

10.1.5 LoginProcessor.java

```
/*
 * LoginProcessor.java
 *
 * Created on 29 June 2012, 18:00
 *
 */

package w2asvb.processor.authentication;

import java.rmi.RemoteException;
import java.util.Collection;
import javax.ejb.FinderException;
import samand.exception.AuthenticationException;
import samand.exception.PlatformException;
import samand.profile.Employee;
import samand.services.EJBReferences;
import samand.services.ResourceRegisterar;
import samand.util.Constants;
import samand.util.DEBUG;

/**
 *
 * @author Ather Mughal
 */
public class LoginProcessor {

    /** Creates a new instance of LoginProcessor */
    public LoginProcessor() {
    }

    public Employee authenticateEmployee(String userName, String password) throws
AuthenticationException, PlatformException{
        DEBUG.println(DEBUG.Debug_Level, "LoginProcessor", "authenticateEmployee()",
"Start");

        //1. Locate employee details in database
        orms.employee.EmployeeRemoteHome employeeHome =
(orms.employee.EmployeeRemoteHome)
                    ResourceRegisterar.lookupResource(EJBReferences.EMP_HOME_NAME);

        //2. Create and load Employee object
        orms.employee.Employee employeeRemote;
        samand.profile.Employee emp;


        /*********************************************************************/
        try {
            employeeRemote = employeeHome.findByUserName(userName, password);

            //3. Return Employee Object

            emp = new Employee();
            emp.setEmployeeID(employeeRemote.getEmployeeId());
            emp.setUserName(userName);
            emp.setFirstName(employeeRemote.getEmployeeName());
            emp.setSurName(employeeRemote.getEmployeeSurName());
            emp.setCompanyAdmin(false);
            //emp.isCompanyAdmin()//hamed: need to about method to verify this role

        }catch (FinderException e) {
            DEBUG.println(DEBUG.Tracing_Level, "LoginProcessor",
"authenticateEmployee()", "Exception Occured[" + e + "]");
            throw new
AuthenticationException(Constants.INCORRECT_USERNAME_PASSWORD_MSG);
        }catch (RemoteException e) {
            DEBUG.println(DEBUG.Tracing_Level, "LoginProcessor",
"authenticateEmployee()", "Exception Occured[" + e + "]");
            throw new PlatformException(Constants.PLATFORM_EXCEPTION_LOGIN_ERROR_MSG);
        }
```

```
        /*********************************************************************/

        /*////////////////////////////temporary creation of employee object
        emp = new samand.profile.Employee();
        emp.setEmployeeID("1");
        emp.setFirstName("Scott");
        emp.setSurName("Tigher");
        emp.setUserName(userName);
        emp.setCompanyAdmin(false);
        ////////////////////////////////////////////////////////////////////*/
        DEBUG.println(DEBUG.Debug_Level, "LoginProcessor", "authenticateEmployee()",
"End");
        //4. Returning Employee Object
        return emp;

    }// end of authenticateEmployee()

    public void authenticateLocation(String ip) throws PlatformException,
AuthenticationException{
        DEBUG.println(DEBUG.Debug_Level, "LoginProcessor", "authenticateLocation()",
"Start");

        /*******************************PLATFORM CODE*********************/
        //1. Locate location details from platform
        orms.location.LocationRemoteHome locationHome =
(orms.location.LocationRemoteHome)
                    ResourceRegisterar.lookupResource(EJBReferences.LOC_HOME_NAME);

        orms.location.Location locationRemote;

        //2. Create and load Location object
        try {

            Collection locations = locationHome.findByIp(ip);
            DEBUG.println(DEBUG.Debug_Level, "LoginProcessor",
"authenticateLocation()", "Location ["+ ip +"] is an authenticated location.");
        } catch (FinderException e) {
            DEBUG.println(DEBUG.Tracing_Level, "LoginProcessor",
"authenticateLocation()", "Exception Occured[" + e + "]");
            throw new
AuthenticationException(Constants.UNAUTHORIZED_LOCATION_ACCESS_MSG);
        } catch(RemoteException e){
            DEBUG.println(DEBUG.Tracing_Level, "LoginProcessor",
"authenticateLocation()", "Exception Occured[" + e + "]");
            throw new PlatformException("Unable to load data from Location Object");
        } catch(Exception e){
            DEBUG.println(DEBUG.Tracing_Level, "LoginProcessor",
"authenticateLocation()", "Exception Occured[" + e + "]");
            throw new PlatformException("Unknow Platform Exception Occured");
        }
        DEBUG.println(DEBUG.Debug_Level, "LoginProcessor", "authenticateLocation()",
"End");
        /*********************************************************************/
    }//end of authenticateLocation()

}
```

10.1.6 WebRequestBrorker.java

```
/*
 * WebRequestBroker.java
 *
 * Created on 29 June 2012, 18:03
 *
 */

package w2asvb.broker;

import samand.context.RequestContext;
import samand.context.ResponseContext;
import samand.context.WebRequestContext;
import samand.context.WebResponseContext;
import samand.exception.AppException;
import samand.processor.IActionProcessor;
import samand.profile.ActionEntry;
import samand.util.DEBUG;
import samand.view.UIView;

/**
 *
 * @author Ather Mughal
 */
public class WebRequestBroker implements IRequestBroker{

    private boolean busy;

    /** Creates a new instance of WebRequestBroker */
    public WebRequestBroker() {
        init();
    }

    public void init(){
        DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "init()", "Start");

        setBusy(false);
        DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "init()", "Initialized
with BUSY STATUS:" + isBusy());
        DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "init()", "End");
    }

    public ResponseContext brokeRequest(RequestContext requestContext){
        DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "brokeRequest()",
"Start");

        //initializing local variables
        samand.profile.Company companyProfile = null;
        samand.profile.Location location = null;
        WebRequestContext webReqCtx = null;
        ActionEntry actionEntry = null;
        IActionProcessor actionProcessor = null;
        String actionProcessorName = null;

        //WebResponseContext webResponseCtx = new WebResponseContext();


        //casting back RequestContext into WebRequestContext
        webReqCtx = (WebRequestContext)requestContext;

        //allocating references including retieval of company profile from context
object
        location = webReqCtx.getLocation();
        companyProfile = location.getCompany();

        //mapping user action into appropriate Action Processor
        actionEntry = companyProfile.getActionMapEntry(webReqCtx.getSourcePage());
        DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "brokeRequest()",
"ActionEntry for this request is ->" + actionEntry.toString());
                //loading the appropriate Action Processor
```

```
        actionProcessorName = samand.util.Constants.DEFAULT_PACKAGE_NAME
+".processor." + actionEntry.getActionProcessor();
        DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "brokeRequest()",
"Loading... ["+  actionProcessorName +"]");

        try{
            actionProcessor = lookupActionProcessor(actionProcessorName) ;

            //executing action processor method
            DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "brokeRequest()",
"Executing...["+  actionProcessorName +"]");
            UIView tempView = actionProcessor.execute(webReqCtx);

            //Setting up reference of UIContainer.UIView
            DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "brokeRequest()",
"Setting up retrieved UIView into UICotainer.");
            webReqCtx.getUiContainer().setUiView(tempView);

            //In Case of sucessfull execution of action
            webReqCtx.appendActionOutput(actionEntry.getDestination());

        } catch(AppException e){
            DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "brokeRequest()",
"System Exception Occured::" + e.getMessage());
            webReqCtx.appendActionOutput(actionEntry.getErrorPage());
            webReqCtx.setAppException(e);
        } catch(Exception e){
            DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "brokeRequest()",
"System Exception Occured::" + e.getMessage());
            webReqCtx.appendActionOutput(actionEntry.getErrorPage());
            webReqCtx.setAppException(new AppException(e.getMessage()));
        }

        DEBUG.println(DEBUG.Debug_Level, "WebRequestBroker", "brokeRequest()", "End");

        return null;
    }

    public void brokeResponse(RequestContext requestContext, ResponseContext
responseContext){
        WebRequestContext webRequestCtx = null;
        WebResponseContext webResponseCtx = null;

        //casting back requestContext into web request context
        webRequestCtx = (WebRequestContext)requestContext;
        webResponseCtx = (WebResponseContext)responseContext;

        //setting up values
        webRequestCtx.setUIView(webResponseCtx.getUiView());
        webRequestCtx.setAppException(webResponseCtx.getAppException());
        webRequestCtx.setActionOutput(webResponseCtx.getActionOutput());

    }

    public void destroy(){

    }


    public boolean isBusy() {
        return busy;
    }

    public void setBusy(boolean busy) {
        this.busy = busy;
    }

    public void release(){
        setBusy(false);
    }
```

```
    //Future Suggestion: Inside this method, action processor can be lookedup
    //using JNDI type Resource Directory, instead of fresh initialization
    private IActionProcessor lookupActionProcessor(String actionProcessorName){
        try {
            return (IActionProcessor)Class.forName(actionProcessorName).newInstance();
        } catch (Exception e) {
            new AppException("WebRequestBroker.lookupActionProcessor()@Unable to load
Action Processor.");
            return null;
        }
    }
}
```

10.1.7 JSFActionDispatcher.java

```
/*
 * JSFActionDispatcher.java
 *
 * Created on 29 June 2012, 19:08
 *
 */

package w2asvb.system;

import samand.SystemManager;
import samand.broker.WebRequestBroker;
import samand.context.WebRequestContext;
import samand.exception.AppException;
import samand.util.DEBUG;

/**
 *
 * @author Ather Mughal
 */
public class JSFActionDispatcher {

    /** Creates a new instance of JSFActionDispatcher */
    public JSFActionDispatcher() {
    }

    //this method contains the common logic of requesting request broker from system
manager
    public void dispatch(WebRequestContext webReqCtx, SystemManager sysManager) {
        DEBUG.println(DEBUG.Debug_Level, "JSFActionDispatcher", "dispatch()",
"Start");
        WebRequestBroker webReqBroker = null;

        //retreving request broker from system manager
        DEBUG.println(DEBUG.Debug_Level, "JSFActionDispatcher", "dispatch()", "About
to retrieve WebRequestBroker from SystemManager.");

        webReqBroker = sysManager.getWebRequestBroker();

        //Broking incoming request to Action processor
        DEBUG.println(DEBUG.Debug_Level, "JSFActionDispatcher", "dispatch()", "About
to broke current request..");
        webReqBroker.brokeRequest(webReqCtx);
        DEBUG.println(DEBUG.Debug_Level, "JSFActionDispatcher", "dispatch()", "Current
request broking completed sucessfully.");
         webReqBroker.brokeResponse(webReqCtx, webResCtx);

        //releasing request broker
        DEBUG.println(DEBUG.Debug_Level, "JSFActionDispatcher", "dispatch()",
"Releasing WebRequestBroker["+ webReqBroker +"].");
        webReqBroker.release();
        DEBUG.println(DEBUG.Debug_Level, "JSFActionDispatcher", "dispatch()",
"WebRequestBroker released.");

        DEBUG.println(DEBUG.Debug_Level, "JSFActionDispatcher", "dispatch()", "End");
    }

}
```