ENTERPRISE COMPONENT ARCHITECTURE FOR MOBILE COMMERCE SERVICES

HLENGIWE PINKY KUNENE

December 2004

ENTERPRISE COMPONENT ARCHITECTURE FOR MOBILE COMMERCE SERVICES

Hlengiwe Pinky Kunene

A dissertation submitted to the Faculty of Science and Agriculture in fulfillment of the requirements for the degree

of

MASTERS OF SCIENCE
in
COMPUTER SCIENCE

Department of Computer Science
University of Zululand

December 2004

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma at any university or other institute of higher learning, except where due acknowledgment has been made in the text.

Hlengiwe Pinky Kunene

Dedication

To my mother Thandi " Mam-B " , my two sisters Sithembile " Gel'teng ", Buhle " Kelly " and my one and only brother Lindokuhle " Boyzin " and to my late father Guduza.

Acknowledgment

Firstly, I would like to thank God for giving me this opportunity to have reached this far. Secondly, my special thanks go to my supervisor Prof. M.O Adigun, who encouraged me to take further postgraduate studies under his supervision. I am grateful for his ideas in making this research a success. Thirdly, my special thanks go to student assistants, who participated in implementing the prototype. These were L.N. Mkhwanazi, P.T. Tamba-tamba, S.C. Chonco, M.B. Linda and M.D. Jojisa. I would also like to give my sincere thanks to my colleagues D.P. Biyela, T.C. Khumalo and M.W. Nkambule for their support. I am furthermore grateful to my friends C.N. Nxele, N.N. Nxele, S.P. Masondo, N. Nkosi for their words of encouragement and support. Lastly, my appreciation goes to all staff members who were always there in giving encouragement, advice and guidance.

Table of Contents

List of Figures

List of Tables

ABSTRACT

This research focuses on creating a component based repository architecture for mobile commerce services called (e-TOCOR) with the emphasis on component storage and retrieval. To realize this framework three tasks were carried out namely (i) a model for engineering component based m-commerce service was defined using existing models (ii) the Universal Description Discovery and Integration (UDDI) was used to model a component repository (iii) the mobile travel reservation application prototype was developed to demonstrate the proposed model. The results obtained were threefold (i) by evaluating the existing component based architectures, the study showed that m-commerce services are not the same as e-commerce services, the Information Requirement Elicitation (IRE) was adopted as a mechanism for eliciting a request and a service delivery protocol for end-user mobile commerce services and (ii) the prototype was developed to show how enterprise components can be delivered in mobile devices using the IRE protocol. It was also shown that the way existing m-commerce services elicit requests takes much time, the shortest way was to use a text message (iii) the repository framework was created emanating from the home-based reference architecture. In conclusion, the proposed repository could not be compared with the existing repository architecture because it was not implemented, instead the UDDI was used.

CHAPTER ONE

1.0 INTRODUCTION

1.1 Overview

Each and every system that is developed has an architecture. The software architecture represents a high level of abstraction from which a system evolves as a collection of interacting components [1]. Different architectural approaches have been defined and implemented.

Traditionally[2], the application architecture consisted of a monolithic system, where the data access, business rules and user interface were combined into a single program. This type of approach worked well for many years, as long as the systems were carefully controlled and all developers used the same programming language and techniques. With advent of the personal computer, the same types of monolithic applications were being built in an environment which was less controlled. The end result of the traditional architecture was minimal reuse of existing code.

The platforms had no support of components and reuse was achieved by source code sharing. Additionally, there was no separation of logical layers in the application which (data, business logic and user interface) could not be separated as distinct elements in the system. Each layer was compiled into a single deployable unit. Some problems with this type of architecture include:

i. the application functionality which could not be reused in another application; and

ii. as the application grew, it became increasingly difficult to debug and maintain.

A different type of application architecture was required for software developers to achieve increased productivity and for systems to achieve reusability, maintainability and scalability [2].

The next evolution in the application architecture is the Component Based Architecture (CBA) which involves building systems in a layered approach. The layers include a data access layer, business layer and presentation layer. The data access layer included all code and logic, which access the data. The business layer consisted of a domain model or services, which encapsulated the business rules of the application. The presentation layer consisted of all user interface related functionality [2]. Components are reusable in other systems. Developing applications from existing components reduce development time and the cost of maintaining components, because a component is developed only once, and it is reused over and over in new and existing applications. Components can be extended to meet new business demands and users. One of the major problem with CBA is the lack of a standard repository infrastructure which allows components to be searched and retrieved for reuse purposes.

The advent of service-oriented architecture, or SOA, resolved the problem of repository in CBA. The architecture provides for services to be stored in a universal service repository similar to the Universal Description Discovery and Integration (UDDI) described in [3]. SOA consists of two layers, namely data access layer and business layer. The presentation layer exists, but is not part of the service-oriented definition. In addition, a service-oriented architecture includes a well-defined service interface which serves as the access point for all external calling applications. The service interface style serves two purposes. Firstly, its method supports a request/response metaphor. Secondly, it is responsible for controlling access to the business layer. Any applications or other services calling the service interface do not have access to the business layer. This provides a loosely coupled structure such that architecture of the service's implementation can be changed without requiring any changes to the calling application [2].

The architectures mentioned above this far do not particularly excel with respect to the peculiarities of mobile commerce. Hence the following assumptions have been made as the basis of this research work:

(1) Firstly, services common to e-commerce applications have been identified in [4] as trading services, workflow services, access control services, event-notification services, user profiling services, data integration services, etc. It is unlikely that these are adaptable to meet m-commerce specific requirements, because some of the services cannot be provided as m-commerce services since they demand more concentration to mobile users.
(2) Secondly, not all the three traditional service categories namely: end-user services, business process services and data services; fit m-commerce which has been found to be

more end-user oriented. Given the limitations of mobile commerce devices, a protocol for providing m-commerce services to end-users is required.

(3) Finally, the Information Requirements Elicitation Technology proposed by Sun et al in [6, 7] is particularly attractive for end-user and service oriented needs of mobile commerce applications.

1.2 Statement of the Problem

Existing distributed and client-serve architectures have a lot to offer to e-commerce. With respect to m-commerce the earlier assumption in the literature that m-commerce is wireless e-commerce have been found to be unacceptable, because m-commerce challenges are different from those of e-commerce.

E-commerce requires that researchers address certain challenges.

In this work, it is required to find a comparison mechanism that brings out the peculiarities of mobile commerce as different from existing e-commerce architecture.

Furthermore a model is to be formulated from a derivative of some existing reference architecture. The model is required to demonstrate the relevance of SOA to the repository infrastructure challenge in CBA.

Finally, it is required that a mobile commerce application is implemented that demonstrates the applicability of the service-oriented repository architecture or model.

1.3 Motivation for the Research

The ongoing "wireless software and service architecture research" in the Department yielded a service-oriented mobile commerce reference architecture. Among the building blocks of the architecture is the technology archetype that serves as the operating environment for other services supported in the architecture. The technology archetype is envisioned as a middleware that provides various utility services such as component repository service, context-awareness and so on. The contribution of this work is to define model repository architecture to be used as the core technology for implementing the middleware in future.

This motivation will not be complete without mentioning the role to be played by the IRE mechanism. IRE has been adopted in this work as the primary protocol for the mobile commerce prototype application built to demonstrate how technology archetype is supposed to work.

1.4 Research goals and objectives

The major goal of this research is to create a component based mobile commerce architecture with emphasis on component storage and retrieval.

The major goal is formulated as an equivalent of some related objectives which are to:

i. find an appropriate model for engineering component based mobile commerce services ;

ii. design a component repository framework that matches the model and

iii. demonstrate how the model is used in mobile commerce using a typical application.

1.5 Research Methodology

The methodology includes:

i. establishing a comparative scheme for relevant architectures via a literature review exercise;
ii. creating a component-based m-commerce architecture from existing models and deriving from it a repository framework and
iii. testing the repository framework via a Mobile Travel Reservation application specifically designed for this purpose.

1.6 Organization of Dissertation

This section describes how the rest of the dissertation is organized. In chapter two a number of fore-runner architectures which were reviewed are explained. A mechanism was created to compare them with the newly formulated architecture in this research.

The focus of chapter three is firstly, the analysis of the repository architecture and secondly the design of the component based mobile travel reservation application that has been used as a proof of concept for the repository framework.

System implementation is the subject matter addressed in chapter four. While a typical application was implemented to provide typical enterprise components, the repository architecture was demonstrated using the public Universal Description, Discovery and Integration (UDDI) infrastructure for web services.

The conclusions and further work are presented in chapter five.

CHAPTER TWO

2.0 BACKGROUND CONCEPT AND LITERATURE REVIEW

2.1 Introduction

Component-Based Process in software engineering, (termed CBSE) is a process of design and construction of systems using reusable software components. It combines concepts from different areas of software engineering and computer science, such as Object-Oriented programming, reuse, software architecture, modeling languages and formal specifications [8].The basic idea of component based development is to qualify, adapt and integrate existing components in that order[9]. Components or a specific business functions in CBSE should be developed in such a way that they can be reused by other systems in the future. Enterprise Components have been defined as software-based specific business functions which can be combined with other components to create a larger system [10]. Components can be provided in mobile devices as mobile commerce services.

Mobile commerce has been defined in [11] as buying and selling of goods and services through wireless handheld devices such as the cellular telephone and personal digital assistants (PDA). Mobile commerce services can be divided into two: information services and transaction services. Mobile users, the users of mobile devices; may use a mobile application to check news, lotto numbers, weather, soccer scores etc. Others may use it to carry out a transaction like, purchasing a book or a concert ticket. Mobile commerce application architectures need to take into consideration the mobile devices limitations such as small screen, slow CPU and limited processing power [12].

The following section presents enterprise component based architectures

2.2 Existing Enterprise Component Based Architectures

2.2.1 Enterprise Business Components (EBC)

Diamelle Technologies' Enterprise Business Components (EBC) [13] comprise a component framework for eBusiness. EBCs were designed and built using the Enterprise JavaBeans 1.1 and they help to provide new business services to customers very quickly, while simultaneously providing a scalable architecture that accommodates future requirements. This architecture is layered, consisting of, Foundation Layer, Component Layer, and Persistence Layer. The Foundation layer provides common base classes for all Enterprise Java Beans and it also provides fine grained building blocks that can be used to build new components quickly for applications. The component layer selectively extends the foundation layer. The layer provides components like user authentication, navigation, view, customer, catalog management, shopping cart, order processing, billing, and shipping. The Persistence Layer is for reuse and flexibility. The purpose of the architecture is twofold: (i) To provide components that are reusable, extendable, and customizable to fit new needs, (ii) To support composition of e-commerce applications. A cursory investigation of EBC architecture shows that it is not service oriented (it lacks service delivery mechanism) and it specifies no rules on how to compose components using this architecture. The framework was implemented using the Java programming language.

2.2.2 IBM San fransisco Framework

IBM's San Fransisco Framework [14] defines three layers of reusable software components. The highest Layer called the core business processes provides business objects and default business logic for vertical domain. The second layer, called the common business objects provides basic facts and rules that are common to most business environments and are used by more than one business process. The lowest layer Foundation Layer provides the object infrastructure that is used to build the Common Business Objects Layer and Core Business Process Layer or to build domain-specific business process. All components within the various layers are highly extensible to support customization and application differentiation. Instead of building the entire application from scratch, application developers can choose to exploit San Francisco Framework at any of the three layers. This architecture supports composition of e-commerce applications.

2.2.3 Service Oriented Architecture

Service oriented architectures [15, 16, 17, 18, and 19] use a component based development approach to develop and compose services. Most of the service-oriented architectures seem to use the same architecture and involve three different kinds of actors: service providers, service requesters and discovery agencies. Usually they provide the same functionality but the difference is in the terms used. The service provider exposes some software functionality as a service to its clients. In order to allow clients to access the services, the provider also has to publish a description of the service. Since the service provider and service requester usually do not know each other in advance, the service descriptions are published via specialized discovery agencies. They can categorize the service descriptions and provide them in response to a query issued by one of the service requesters. As soon as the service requester finds a suitable service description for its requirements at the agency, it can start interacting with the service provider and using the services. Such service oriented architectures are typically highly dynamic and flexible because services are only loosely coupled and clients often replace services at run-time. Service oriented architectures provide services using service discovery protocols. Most of the service oriented architecture such as Jini, use the service location protocol as a way of delivering e-commerce services.

Service oriented architectures use web services technologies. Web service technologies have been defined in [15] as follows: from a technical perspective, web services are a standardized way of integrating web-based applications using open standards including XML, the simple object access protocol (SOAP), the Web Services Description Language (WSDL), and the universal description, discovery, and integration (UDDI) specification. UDDI is a web service that lets businesses discover one another and describe how they interact. It provides a simple object access protocol interface for publishing entries and querying the UDDI registry. Multiple providers can register their services in a central directory and precisely characterize their offering.

2.2.3.1 Common Picture eXchange (CPXe)

CPXe [19] is a highly interoperable service delivery framework that leverages the web services paradigm to give providers access to an expanded market and offer consumers a broad range of digital imaging services. The CPXe architecture consists of three tiers, the service themselves, a directory service, and the applications that discover and interact with these two types of services. It only defines access services for online fulfillment. It relies on the UDDI specification for directory functionality. A service locator mechanism lets consumers easily select vendors offering the products and features they desire. A service locator is implemented to function as a travel agent or sales broker. A service locator consults the UDDI directory to determine available services and queries those services for catalog information. Catalogs in this framework give a provider a standardized way to communicate detailed information about its products and services to a service requester.

2.2.3.2 Jini

Jini [20] is described as a distributed service-oriented architecture developed by Sun Microsystems. Jini services can be realized to represent hardware devices, software programs or a combination of the two. A collection of Jini services forms a Jini federation. Jini services coordinate with each other within the federation. The overall goal of Jini is to turn the network into a flexible, easily administered tool on which human and computational clients can find services in a flexible and robust fashion. Jini is designed to make the network a more dynamic entity that better reflects the dynamic nature of the workgroup by enabling the ability to add and delete services flexibly. One of the key components is the Jini Lookup Service (JLS), which maintains dynamic information about the available services in a Jini federation. A user searching for a service in the network first multicasts a query to find out the Jini Lookup Service in the network. If a Jini Lookup Service exists, the corresponding remote object is downloaded into the user's machine. The user then uses this object to find out its required service. In Jini, service discovery is done by interface matching or java attributes matching. If the Jini Lookup Service contains a valid service implementing the interface specified by the user, then a proxy for that service is downloaded in the user's machine. The proxy is used henceforth to call different functions offered by the service.

2.3 Service-Oriented Mobile Commerce Reference Architecture

The reference architecture shown in figure 2.1 is conceptualized with the tourism sector in mind. A product-line is an architectural concept defined as a collection of systems sharing a managed set of features constructed from a common set of core software assets[21]. The product line architecture consists of five archetypes or architectural elements, namely client, technology, service, information and transaction. Each of these elements are architectural abstractions that have to be instantiated when a product is being derived from the architecture. Each archetype is briefly overviewed as follows:

Client stands for any mobile device that can serve as user interface to the product being instantiated. Examples are PDA, the cellular phone, the handheld PC, etc.

Technology- refers to the operating environment in which services are to run. To ensure services are standardized the environment was conceptualized as a set of utility services through which data, metadata and context information are served to other architectural elements according to the IRE protocol (depicted in figure 2.2). This archetype is the basis of the proposed repository architecture in this work envisioned as a middleware, it consists of four system services which other services (must adopt information, transaction, even third party services).

A service is an abstract element that will not be instantiated but serves as the generic archetype for technology, information and transaction.

Information refers to all content-based services to be requested by the client.

Transaction models all commercial activities services in which there is an exchange of money and product.

It is to be noted that the architecture further identifies both information and transaction archetypes as product-line services in contrast to third-party services which are loaded from external sources. An example of Third party services may be a game downloaded from the internet or similar network service.

In order to realize the technology archetype as a set of utility services, a protocol or operating environment style was needed. There is no existing standard for mobile commerce in this respect, so the Information Requirement Elicitation (IRE) Technology, first advanced by Sun et al [6,7] is our choice for driving the reference architecture.

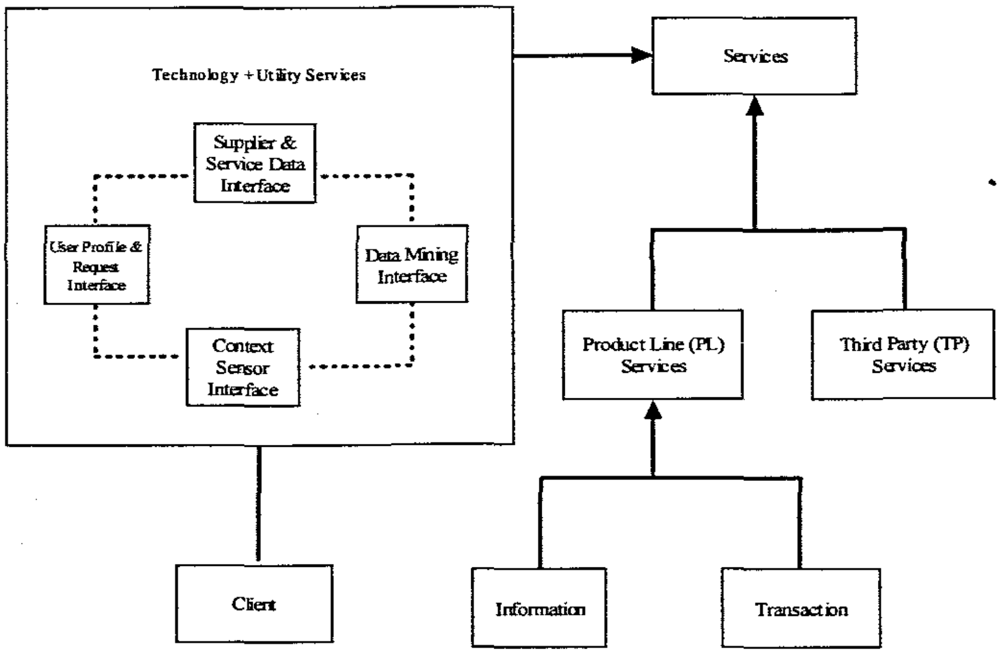

Technology + Utility Services
Supplier & Service Data Interface
User Profile & Request Interface
Data Mining Interface
Context Sensor Interface
Services
Product Line (PL) Services
Third Party (TP) Services
Client
Information
Transaction

Figure 2.1: M-Commerce Service-Oriented Reference Architecture

2.4 Proposed Model

This research proposes to create a layered reusable component based repository framework for mobile commerce services called eTOCOR (e-Tourism Component Repository) emanating from the product line reference architecture of the previous section. The proposed architectural framework will be used to compose applications. Applications are composed of components.

In CBSE , a component is specified in terms of its functional and extra-functional properties[22]. In this research the focus is on the functionality of a component. The functionality of a component is accessed via an interface [23]. The main aim is to create a reusable architecture that can be used to compose mobile commerce service.

Mobile devices have limitations such as small screen, limited processing power, the device size and the input device such as small buttons An IRE protocol was adopted as a request/response mechanism to invoke service or to elicit a request from the mobile user. Tourism have been chosen as the domain of sample enterprise components to be used in mobile commerce. The reference architecture is designed in such a way that product instances developed using the architecture are IRE-enabled. An IRE-enabled component is a component that uses IRE protocol (see Figure 2.2) to elicit a request and provide services according to a user's request. Therefore, IRE enablement consists of conformity to the IRE driven utility service specification of the technology archetype (Fig 2.1)

2.5 A Framework for Comparing Architectures

In Table 2.1, an attempt has been made to compare existing component-based frameworks and the component repository (eTOCOR). The comparison is based on five characteristics namely: Layering flavour, core technology, architectural style, service orientation, and service discovery protocol (if relevant). It is to be noted that most of the popular architectures are component based and electronic service oriented but are not suited for mobile commerce model.The main deficiency in the existing architectures is the absence of the likes of IRE which is a service discovery protocol specifically designed for m-commerce.

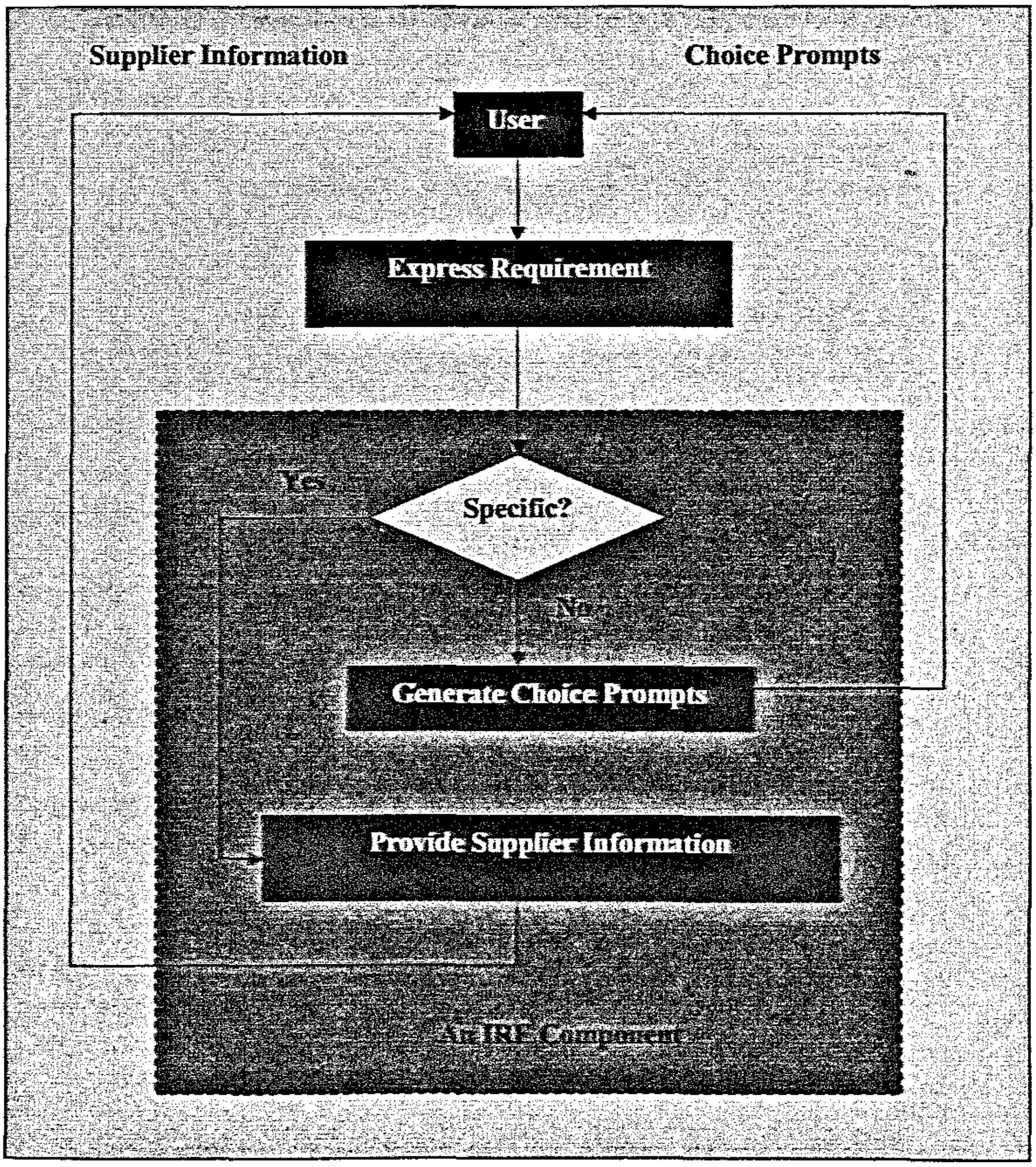


Figure 2.2 Overview of the IRE protocol as in Sun et al [7]

Table 2.1 : Comparison of the proposed architecture (eTOCOR) and existing architectures

Item being compared	**Layering Flavour**	**Core Technology**	**Architectural style**	**Service Orientation**	**Service discovery protocol**
Enterprise Business Components[13]	Consist of component layers -Foundation layer -Component layer -Foundation layer	Enterprise JavaBeans 1.1	Popular e-commerce patterns	e-commerce services	-
IBM San Fransisco Framework[14]	Consist of component Layers -Core business process layer - Common business layer -Foundation layer	Java Technology	Design Patterns(Gamma el al)	e-commerce services	-
Common Picture eXchange (CPXe) [19]	Consist of three tiers, -services themselves, -directory services, - applications .	eXtensible Markup Language (XML)	Service Oriented Architectural style	e-commerce services	Service Location Protocol (SLP)
Service Oriented Architecture(SOA) such as Jini Architecture [15,16,17,18]	Consist of layers -business layer - data layer -presentation layer	Java Language	Service Oriented Architectural style	e-commerce services	Service Location Protocol (SLP)
***e-Tourism* Component Repository (eTOCOR)**	three layers, - repository layer - application layer, - presentation layer	Java Technology and Web Services Technologies	Product line style	*m*-commerce service	Information Requirement Elicitation (IRE) Protocol

CHAPTER THREE

3.0 REPOSITORY ARCHITECTURE ANALYSIS AND DESIGN

3.1 Introduction

Deri [24] defines and differentiates between an architecture and a framework. An architecture is defined by means of a framework, and it specifies and restricts the way components interact. He further describes architecture as a conceptual description of a system presented at a level of abstraction in which the system's high-level design can be understood. A repository architecture specifies:

i. the structure of the repository, the services it provides, and its responsibilities with respect to other archetypes in the reference architecture;

ii. the repository interface, consists of component characteristics that have to be visible from the outside and

iii. the way constituent components collaborate.

3.2 The Repository Framework : eTOCOR

The architectural framework shown in figure 3.1 provides the context in which the repository architecture is designed. It consists of three layers the repository layer, application layer and the presentation layer.

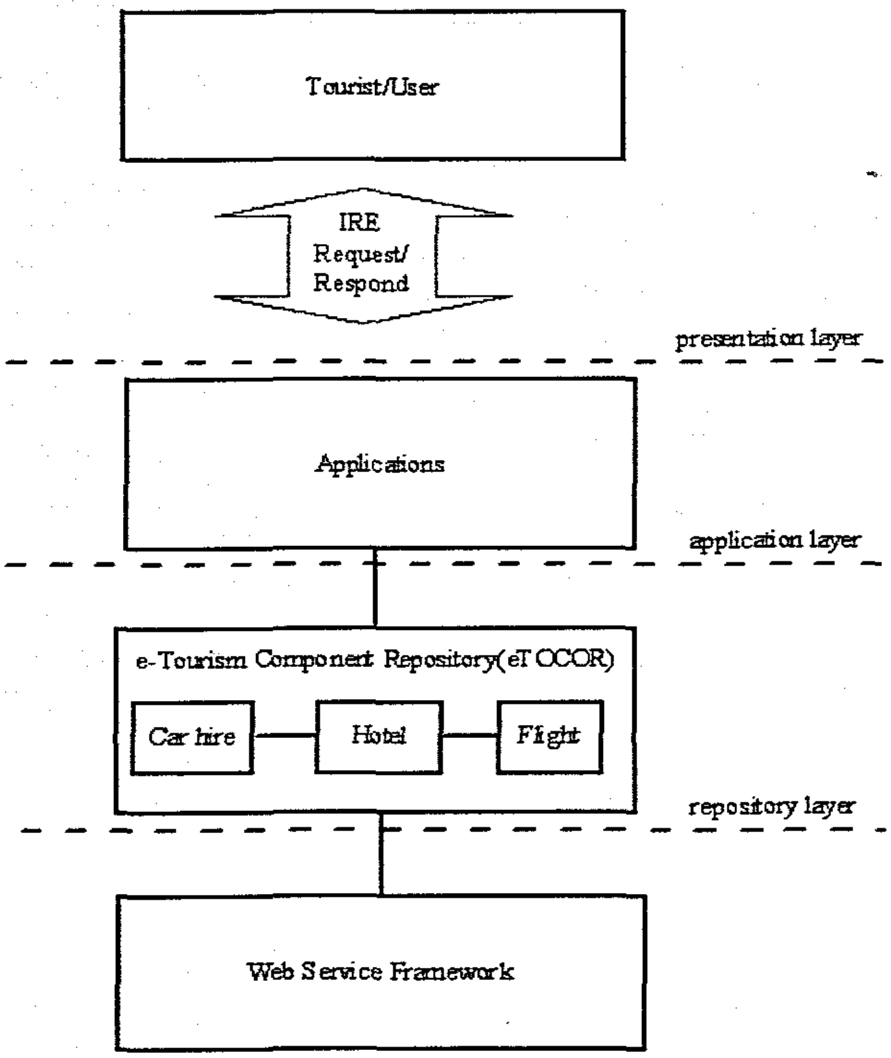


Figure 3.1 Architectural Context of the Repository Framework

3.2.1 The Repository layer

The repository relies on the web service technologies, Universal Description, Discovery, and Integration specification (UDDI) for directory functionality. It provides interfaces and operations for storing, retrieving and updating components. A component can be identified by a name. A component structure contains information about the component. It includes the name of the component, the description of a component and the service it provides. Components are categorized according to services they provide. Each component has its own interface and describes its services.

Components in this work are designed to be reusable to other applications within the tourism domain. eTOCOR components used the Java Servlet Technology to describe the client interface. The eTOCOR components are published and written in WSDL in a registry (UDDI) or made available for discovery. When another enterprise component needs a component that satisfies some functionality, then a query is issued to the repository to find if WSDL of an available component exists. A component is a shared resource, which means the component can support many applications. This type of flexibility is needed because many different applications can share a single component. A component is to be located using the component repository.

3.2.2 The Application layer

eTOCOR-based application makes services available on mobile devices. Internally the application is composed of components. An application specifies a mechanism for invoking a service. In an eTOCOR component are tourism services. Examples are car hire service, hotel reservation service, flight booking services. Each and every application incorporates standard elements such as end-users, services, execution environment, and communication models. End-users refer to mobile users e.g. Tourist or Service Requester using a mobile device to request a service. Services are provided by applications that are available on the network e.g. hotel reservation service, etc. Execution environment: an environment where an application runs. The communication style, which refers to communication between a user and applications, is called “request and respond”. Thus a user of an application issues a request and the application responds to the request. eTOCOR applications use on the client-server communication model.

3.2.3 The Presentation Layer

The presentation layer refers to the user interface where users, service requesters or tourists can interact with a service. It provides an access point to services offered by an application. An eTOCOR application provides a message box interface for requesting a service using a structured text message. Service requesters are able to issue a request using a message, for instance, to request a reservation service such as flight, hotel or car. Each service has a message format depending on the service to be invoked.

To instantiate reference architecture, the tourism domain was selected. A mobile travel reservation application was developed to demonstrate IRE enabled components crafted in the mobile commerce repository framework called eTOCOR. IRE is an elicitation mechanism; hence a text message was selected as a means of eliciting and responding to a service request. The way e-commerce services are delivered on desktop machine contrasts with m-commerce service delivery on hand-held devices. The latter adapts to the smaller screen of the hand-held device.

Each component is IRE-enabled to be used in a mobile device as follows, illustrating the Travel Reservation components

i. Tourists were required to register to personalize their wireless devices on a web application;

ii. Once the tourist was registered he/she was able to request services according to his/her preferences;

iii. The tourist used a message to request a service or to make reservation;

iv. Once the tourist had sent his/her request, his/her preferences were used to deliver services and

v. The confirmation number was issued.

Services are thus requested using a text message on a mobile phone. The message conforms to a format depending on the service to be requested. Most messages are formatted using abbreviation. In this case reservation messages were formulated based on the format represented below:

Hotel <htl> <destination_city> <check_in_date> <check_out_date>

Flight <flt> <departure_city> <arrival_city> <reservation_date>

Car <car> <pick_up_city> <drop_off_city> <reservation_date>

For example to request a hotel reservation service , a message such as this; <htl> <dbn> <03/12/04> <04/12/04> where <htl> stands for hotel, <dbn> stands for Durban, <03/12/04> stands for check-in-date and <04/12/04> stands for check-out-date.

3.3 Component Architecture Design

A structured approach or analysis and design method suggested by Zimmermann et al[26] was used to develop components of the repository architecture (eTOCOR).

The design and implementation consists of the following activities: Tourism Domain Analysis, Domain Decomposition, Goal Service Model Creation , Subsystem analysis, Service allocation and Component Specification

The top down aspect of this approach came from taking business perspectives and models into consideration: Business functions, processes, sub-processes, and use cases were elaborated to form the outlines of component boundaries. Components provide boundaries and containers for services often discovered through use case analysis and goal-service model creation [27]. The following sections outline the tasks taken to develop the component based repository architecture. Each of the tasks is applied to the tourism domain.

3.3.1 Tourism Domain Analysis

Domain Engineering is defined as a process of defining the scope (i.e., domain definition), analyzing the domain (i.e., domain analysis), specifying the structure (i.e. domain architecture development), and identifying components that will support reuse[28]. It can also be defined as the activity of understanding, abstracting and modeling a bounded problem domain, the people (roles) involved and its enterprise context [29]. The requirements of the component system are analyzed by refining and structuring them. The purpose of doing this was to achieve a more precise understanding of the requirements and to achieve a description of the requirements that is easy to *maintain and that helps us give structure to the whole component application including its* architecture. The major effort in requirements is to develop a model of the application that is to be built, and the employment of use cases is an appropriate way to create such a model. Use cases [30] offer a systematic and intuitive way to capture the functional requirements with particular focus on the value added to each individual user or to each external application. Their key role in driving the rest of the development work has been an important reason for their acceptance in most approaches to modern software engineering.

This framework supports the Tourism specific domain. Mekonnen in [31] defines tourism as "travel and stay of a non-resident." In order to travel to a particular area, there must be a reason. For example, a person or a tourist may travel for leisure, business, visiting friends and relatives, health, education, etc. The tourist chooses a destination for one or the other reason. Transport is necessary to travel and accommodation to stay at the destination. Tourism is a service-based industry comprising a number of related activities, some of which are:

i. Attractions{arts & craft, natural attraction, conference};

ii. Accommodation{ hotels, motel, guest houses, caravans};

iii. Transport{air, water, surface};

iv. Scenic{parks, beach};

v. Entertainment{cinema, theatres, video games, festivals/concerts};

vi. Cultural/Tradition{museums, religion, historical places} and

- Others{news, weather}

Figure 3.2 shows the various interacting elements in the tourism enterprise. Role players in the tourism domain are referred to as Tourist or service requesters, service suppliers examples are hotel suppliers, car suppliers, flight supplier's, etc. Service suppliers provide services which were classified into attraction, accommodation, scenic, transport, and entertainment. To make use of tourism services, Tourists are required to make reservation before arrival. To take a few services such as flight, car, hotel, etc. These services are classified under the accommodation and transport activities. These services share common features such as make reservations, change reservations and cancel reservations which makes them reusable. (see table 3.1).

Services	Common Features
Hotel	Make reservation
	Cancel reservation
	Change reservation
Car	Make reservation
	Cancel reservation
	Change reservation
Flight	Make reservation
	Cancel reservation
	Change reservation

Table 3.1 Tourism services and common features

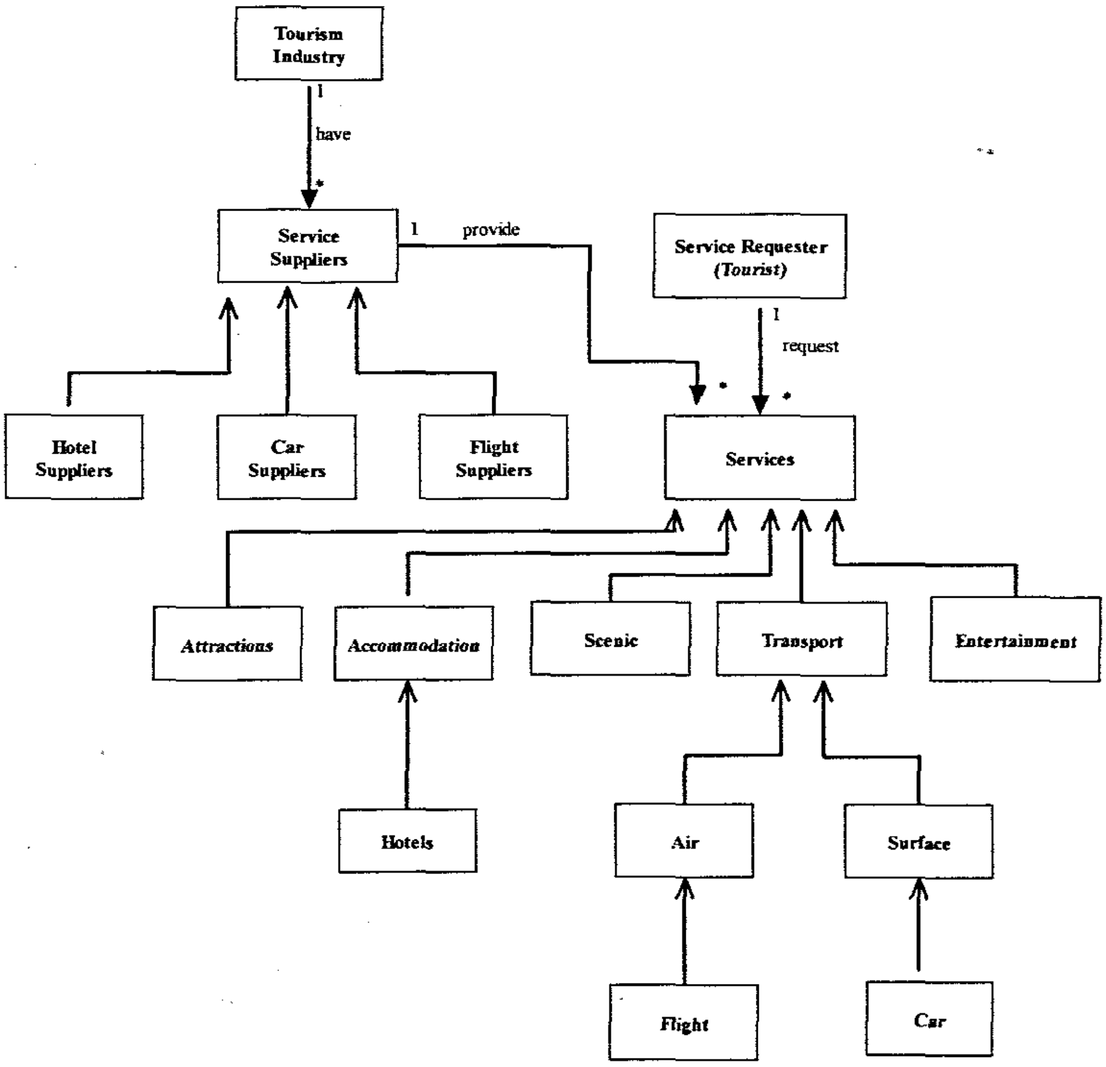


Figure 3.2 Tourism Conceptual Model

3.3.2 Domain Decomposition

In this task, the domain was decomposed into business processes, sub-processes and use cases. From a business perspective, the domain consists of a set of functional areas [27]. As a result of the decomposition, the following functional areas were obtained:

i. Travel Reservation Business Process;

ii. Customer Relationship Management Business Process;

iii. Payment Business Process and

iv. Marketing Business Process

After decomposing domain into functional areas, each one was decomposed into sub-processes and business use cases.

Use Case Model

Use cases were used to show the functionality of components from the actor's point of view. Figure 3.3 shows a use case model, using a Unified Modeling Language (UML)[32]. The decomposition of the functional areas described above led to the following set of business use-cases:

i. Advertise;

ii. Place order;

iii. Make reservations;

iv. Register and

v. Make payments.

The business use case definitions are business driven and offer common, reusable business functionality [27]. The use case model is explained as follows:

Advertise- this process is activated by tourism service supplier to advertise their product or services to tourist or service requesters.

Place order – this process is activated by tourist or service requesters, to purchase any product or to order any tourism product.

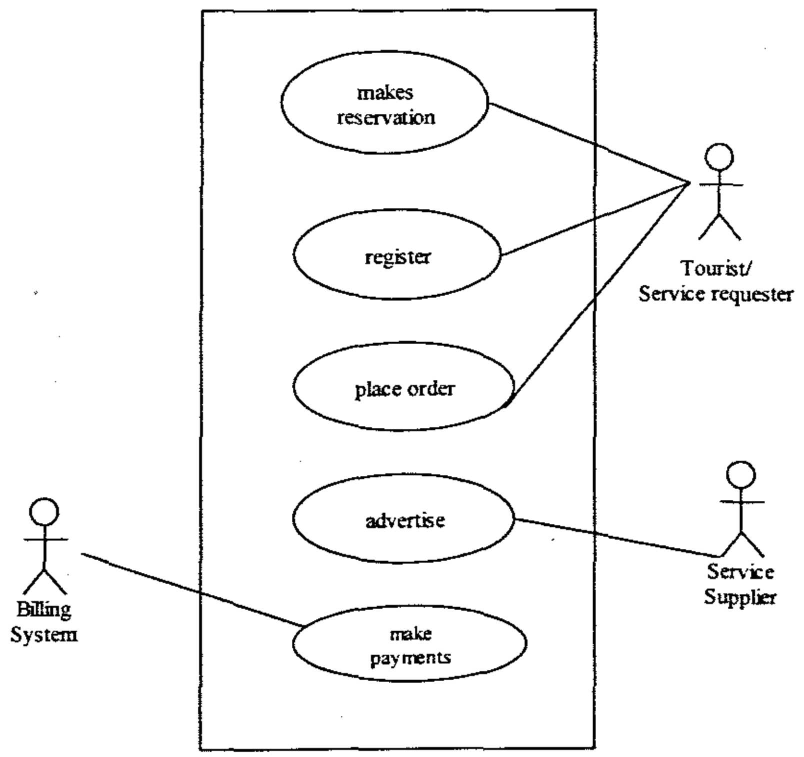


Figure 3.3 Use Case Model for Tourism Domain

Make reservations – this process is activated by a tourist to make a car reservation, make a hotel reservation or to make a flight reservation. These subsystems were given the generic name, make reservation subsystem.

Register- this process is activated by a tourist to register his/her personal details and to set his/her preference.

Make payments – this is done after the requester's request is successful and a payment is made to a supplier who supplied the service.

As a result of tourism domain decomposition, some functional areas have emerged. These are then matched with business patterns as documented in the table 3.2 The business patterns shown in table 3.2 are mostly used in enterprise architectures and they are defined in [27] as follows: The end-user services business pattern allows a tourist to interact with business services. The extended enterprise business pattern allows one business to interact with another business service. In this work the focus is on end-user services.

Not all use cases are covered. Only the make reservation, register use case and make payment use cases are demonstrated. These business use cases were further analyzed to decompose the domain. The business use cases constitute the component in the tourism domain that can be used to compose mobile applications and serve as mobile commerce services.

As we moved to the design, each functional area was mapped to one or more subsystems. Subsystems are pictured as technology services of a business.

Table 3.2 Business Use Cases and their associated patterns

Use Case Name	Description	Requester/ Invoker	Implemented by	Business or Intergration Pattern
Place order	Tourist purchase goods by placing an order	Tourist/ Service Requester	Service Supplier	End-user service
Make Reservation	Tourist makes reservation e.g car, flight, hotel	Tourist/ Service Requester	Service Suppliers	End-user service
Advertise	Service Supplier advertise services	Service Supplier	Service Supplier	Business Pattern
Register	Tourist register mobile device	Tourist	Service supplier	End-user service
Process Payments	service supplier	Service Supplier	Third Parties	Application intergration and extended enterprise
Sends Confirmation	Service supplier system sends confirmation	Service Supplier system	Service Supplier	Appplication Intergration and extended enterprise

3.3.3 Goal-service model creation

For each enterprise components or the functional area identified in domain decomposition, their services were identified using the goal service model [33]. The goal-service model was created as shown in table 3.3 to identify components and services that the repository supports for mobile commerce services. Various notations are possible for a goal-service model. The table was used to document the goal service model which consists of two columns. The first column provides a goal and the second column are services for each goal. This example covers the customer relationship management, travel reservations, and marketing business functions.

Goal	Services
To automatically allow Tourist to make reservations to their favourite service supplier	Register tourist online
Provide end-user reservation service	Make reservation using a mobile device
To allow Tourist to submit their request to the system	
To automatically return all reservations back to the tourist once processing is complete	Send reservation confirmation number
Make Special offers	Register customers in a loyalty programme
	Create loyalty offering
	Communicate offering to registered customers
Promote Seasonal/Holiday Special packages	Create Holiday Package
	Identify marketing outlets
	Use outlet to distribute

Table 3.3 Goal-Service Model for domain decomposition business functions

The first goal listed in table 3.3, is "To automatically allow Tourist to make reservations to their favourite suppliers" and the associated service is "Register tourist online". To achieve this goal, it is important that end user services should be customized in such a way that services are easily used. Tourists are required to register online, set their preference, so that when delivering services in mobile device, services are delivered according to their preferences. The second goal "To provide end-user reservation service", this goal is achieved by allowing tourists to access services using their mobile devices. Tourist interacts with the service through mobile device interface. Tourists are required to submit their request to the mobile application using their mobile device. The mobile application components are IRE-enabled. IRE-enabled systems should utilize user preference information to narrow down options by their personal relevancy [7].

3.3.4 Subsystem Analysis

Each of the business processes mentioned in section 3.3.2 were further broken down, to identify the component boundary of each business components. The main tourism business processes or functional areas are marketing, travel reservation, customer relationship management and payments. This partitioning was based on business process boundaries; end-to-end services that form a business processes [33]. The entire business domain is seen to consist of a set of functional areas, each responsible for making a certain set of cohesively related design decisions. For example, marketing is responsible for defining service packages, their offerings, target customer and pricing for each package offered. Marketing is further concerned with "Specials" catalog or set of packaged services that are offered. Customer Relationship Management (CRM) is responsible for managing tourist, Tourist profile (personal details), preferences (favourite's products, services) are services that the tourist enjoys the most when visiting a tourism supplier's site. Travel Reservation Business Process is responsible for reserving hotel rooms or reserving flight or hiring a car. A typical use case grammar learned from [33, 34] was used to specify the business component boundaries of the four business components given below.

1. Marketing = {Service Packages, "Specials" Catalogs, Pricing}

2. Payment = {bill [services], Payment {transaction process}}

3. Travel Reservation = {[identification], Reservation Request, Reservation Process, Payments, Confirmation}

4. Customer Relationship Management= {Contact Management (Address), Customer profile and Preferences}

We further identified the required functionality for each business component; the system level use cases for each component. For each use case, a sequential diagram was associated with business component use case. Customer Relationship Management (CRM), this component allows Service Requesters to register their mobile device by registering their personal details, set travel preferences such as hotel preferences, car preferences, and flight preferences. The CRM component is associated with each reservation component such as flight, car, and hotel.

When the tourist requests a service, it becomes easy because all the user's preferences are known from the Customer Relationship Management component. During the registration process shown in figure 3.4, the tourist sets his preferences by selecting three preferred hotels. These preferences are used when a tourist requests a service. When a tourist requests a service shown in figure 3.5, he/she writes and sends a message. The hotel reservation component is activated, it then communicates with the customer relationship management component to get the tourist preference. The reservation components check the first preference if there are any services available during the date specified by a tourist. If there is a room available, the room is reserved and the confirmation number is issued together with the hotel name. If all preferences are checked and find that there are no services available during the date specified by the tourist, a list of hotels is presented that have accommodation service available during the date specified by the tourist.

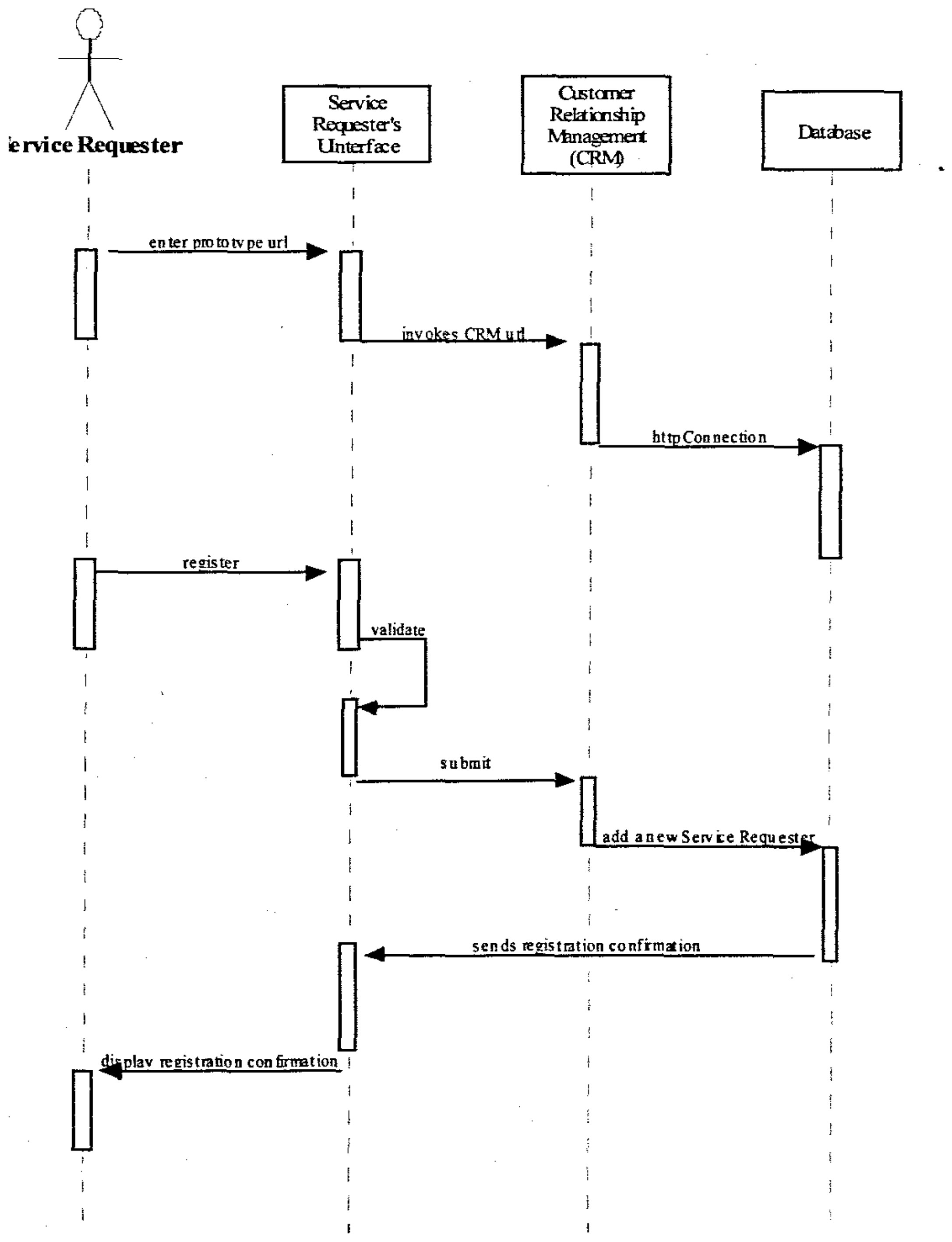

ervice Requester
Service Requester's Unterface
Customer Relationship Management (CRM)
Database
enter prototype url
invokes CRM url
httpConnection
register
validate
submit
add a new Service Requester
sends registration confirmation
display registration confirmation

Figure 3.4 A Sequence Diagram for a Register Use Case

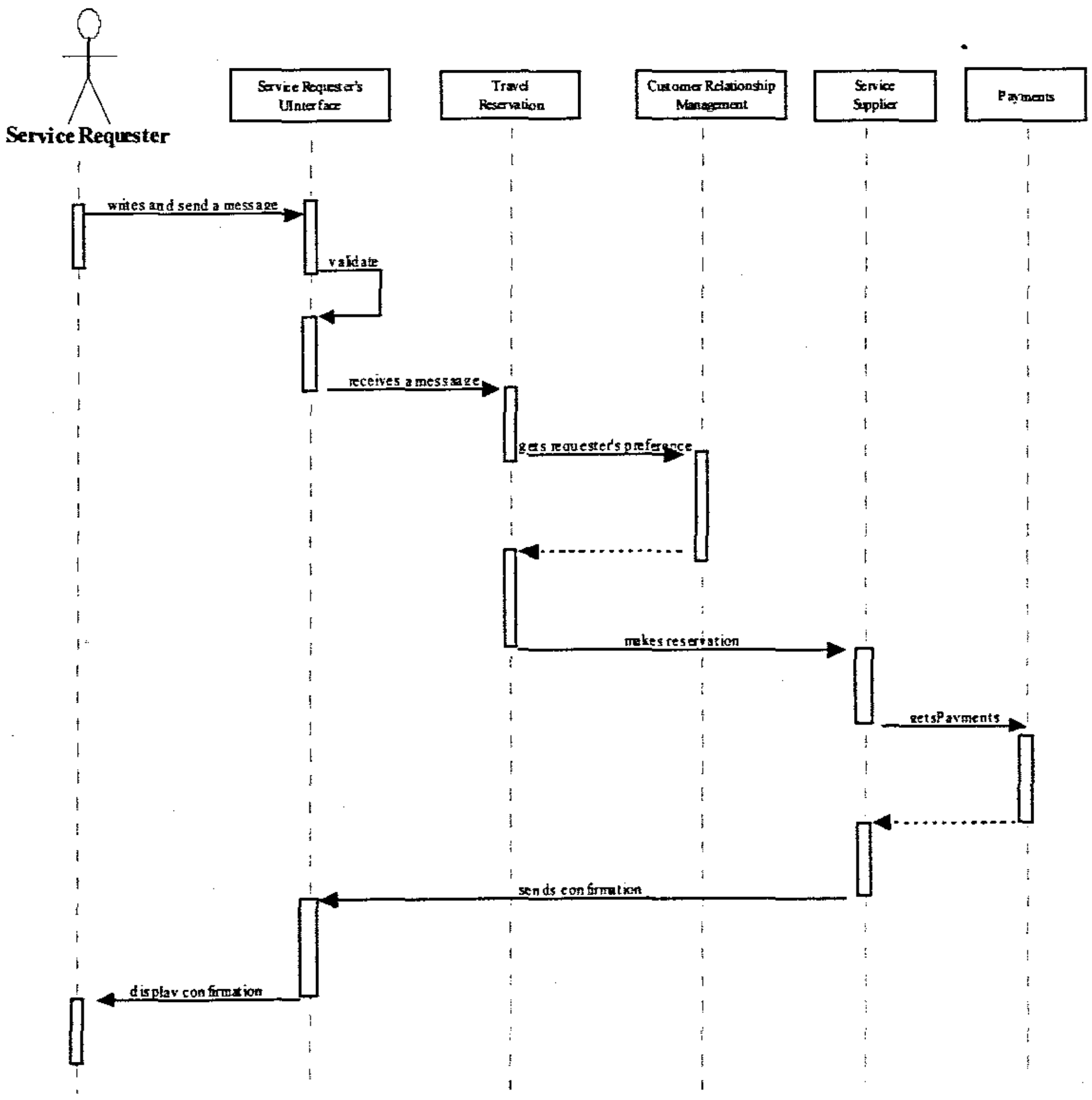


Figure 3.5 A Sequence Diagram for Make Reservation Use Case

The tourist selects one hotel. Then after the reservation is made, payments can be made to the service supplier. The same procedure is used to request a car reservation and flight reservation service.

3.3.5 Service Allocation

Services were identified through a combination of domain decomposition and goal-service modeling shown in table 3.3. Each and every service has its own component where it is contained. The business processes or functional areas are components which contains services.

3.3.6 Component Specification using Use Case Grammar Specification

Enterprise Components were defined around business processes boundaries and often encapsulate a set of related use cases. The progression was seen to be "business process → subsystem → enterprise component [34]". This brought us to the design of the internal structure of enterprise components as in the one which consists of the application of subsystem analysis, along with domain specific languages to define enterprise-scale, loosely coupled business service component and their interfaces. Figure 3.6 shows how business grammar learned in [34] was used to create a domain-specific vocabulary for the Mobile Travel Reservation Enterprise Component.

Mobile Travel Reservation = {[identification], Reservation Request, Reservation Process, Payments, Confirmation}

Identification = {Challenge User with Login, Verify Mobile Number and Password}

Reservation Request = {Write message, Validate, Send message }

Write Message = { hotel(htl), destination city, reservation date |flight (flt) departure city , arrival city, reservation date | car pick-up city, drop-off city, reservation date |vacation package, origin city, destination city, reservation date}

Validate = {validate if the message is in the valid format or not}

Send Message = { press submit}

Reservation Process = {Check customer preference, check service availability, Payments, generate confirmation number, send confirmation number}

Payments = {Transaction process}

Confirmation ={ Sends a confirmation number to a service requester, Display confirmation on mobile device}

Figure 3.6 Business Grammar for Mobile Travel Reservation Component

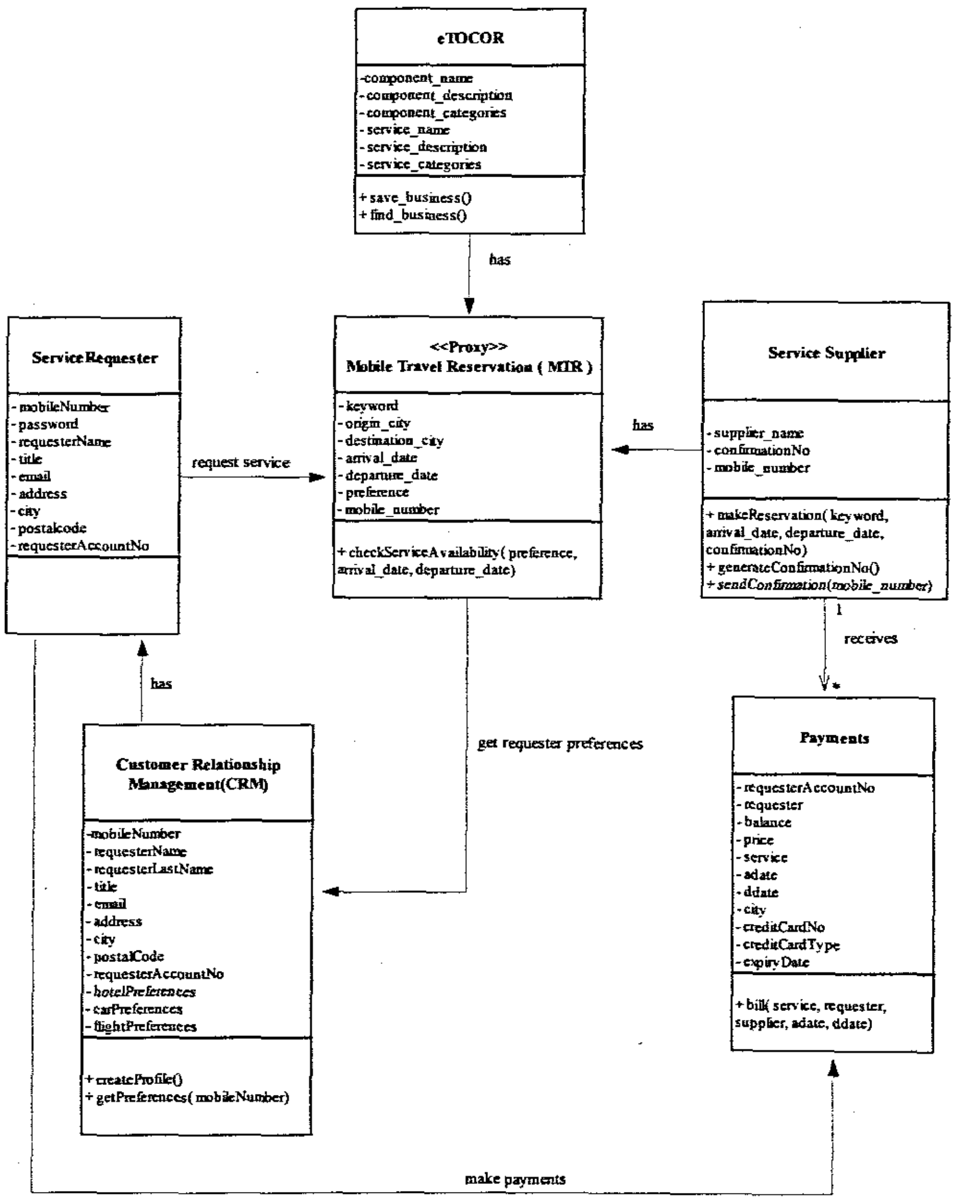


Figure 3.7 Class Diagram for Mobile Travel Reservation Application

A Proxy pattern was used to structure the enterprise components shown in figure 3.7. Gamma et al in [35] defines a proxy as a surrogate or placeholder for another object to control access to it. One reason for controlling access to an object is to defer the full cost of its creation and initialization. The mobile travel reservation application is a proxy, which acts as directory of tourism service components which are contained in the component repository eTOCOR. The mobile travel reservation proxy maintains a reference that lets the proxy access to real components that provide the services. It controls access to the components that provide services and may be responsible for publishing them.

The next chapter implements the prototype mobile travel reservation enterprise component.

CHAPTER FOUR

4.0 IMPLEMENTATION OF THE MODEL

4.1 Introduction

Implementation is the transformation of design into a working program. This chapter presents the model implementation prototype called a mobile travel reservation application and the demonstration of the component repository called eTOCOR, where the mobile travel reservation interfaces were published. The tool, JBuilder 5 environment was used to implement the prototype. It has a built-in Tomcat 3.2.1 web server. The communication protocol between server and the database was done using the JDBC (Java Database Connectivity)[36]. This protocol allows connections to a database, create SQL statements, and run queries. The screenshots shown below were prototyped using Java Servlet. Servlet[37] provides a component based platform independent method for building web-based applications. They have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. They can also access a library of HTTP – specific calls and receive all the benefits of the mature Java language including reusability. They run on a web server which could benefit a mobile device since it has a limited processing power. The prototype shows how enterprise components can be used or can be provided in mobile devices. A rectangle shown in the prototype screenshots emulates the screen of a mobile device e.g a cellphone.

The next section gives an overview of the prototype, and then discusses the partitioning of the application into components. The snapshots of the application prototype are presented and the snippets of implemented functionality are found in the appendix.

4.2 The Mobile Travel Reservation Application

The mobile travel reservation application allows a tourist to make reservation for one or more of flight, car and hotel using a text message. Most end-user applications are user friendly. The IRE was adopted to deliver services to mobile users according to their preferences. The application was composed from the following components: the service supplier, the hotel reservation, the flight reservation, car reservation, customer relationship management, payments and service requester. Figure 4.1 shows application components.

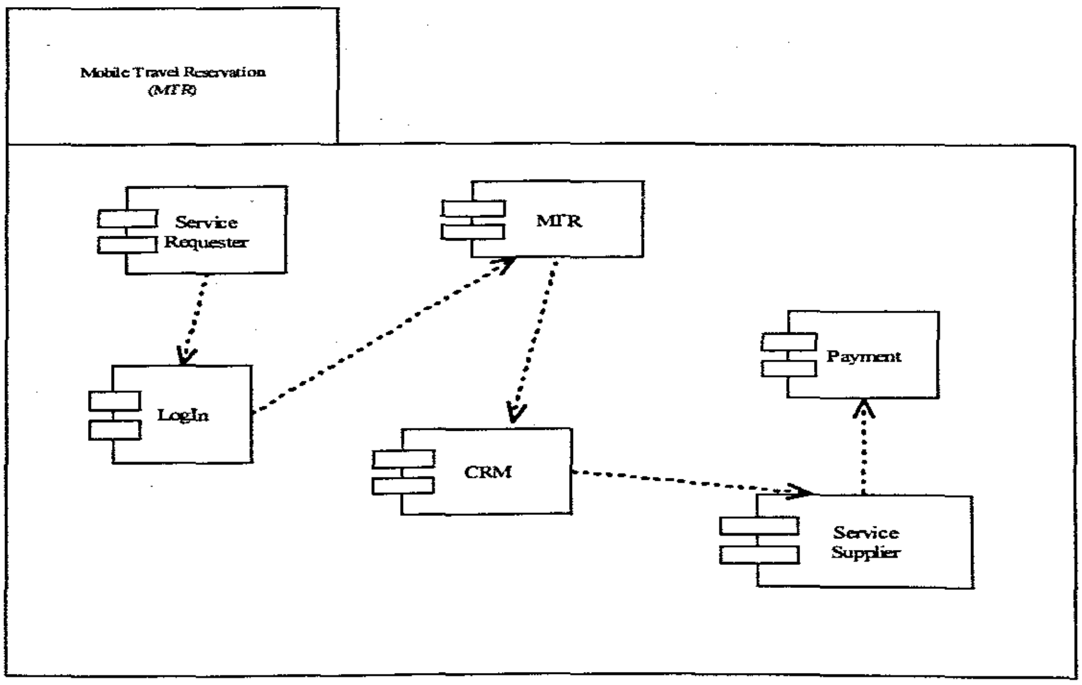
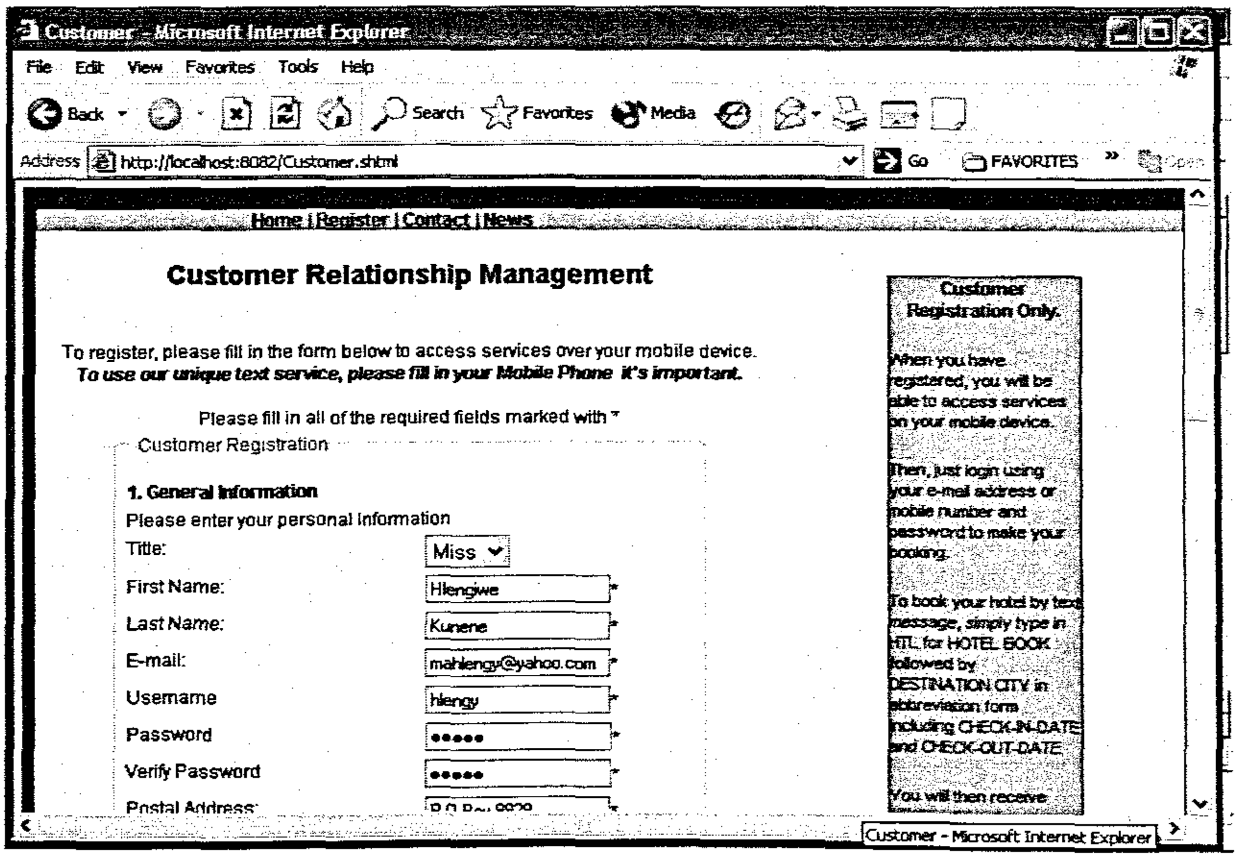


Figure 4.1 Implementation model for mobile travel reservation application

To realize this application these components collaborate with each other such that a component uses the services provided by another component if necessary.

Each of the components could be explained as follows:

The Customer Relationship Management (CRM) – This component was used to create a new user and to capture user profile information. This includes capturing attributes such as the address, email, mobile number and user preference such as hotel preferences, car preferences, and flight preferences. The CRM web application was used to create a user profile. Information that was collected during user registration was associated with the



Customer - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8082/Customer.shtml
Go FAVORITES
Home | Register | Contact | News

Customer Relationship Management

To register, please fill in the form below to access services over your mobile device.
To use our unique text service, please fill in your Mobile Phone it's important.

Please fill in all of the required fields marked with *

Customer Registration

1. General Information
Please enter your personal Information

Title:	Miss
First Name:	Hlengiwe *
Last Name:	Kunene *
E-mail:	mahlengy@yahoo.com *
Username	hlengy *
Password	●●●●● *
Verify Password	●●●●● *
Postal Address:	P.O Box 9939 *

Customer Registration Only.

When you have registered, you will be able to access services on your mobile device.

Then, just login using your e-mail address or mobile number and password to make your booking.

To book your hotel by text message, simply type in HTL for HOTEL BOOK followed by DESTINATION CITY in abbreviation form including CHECK-IN-DATE and CHECK-OUT-DATE.

You will then receive

Figure 4.2 CRM Interface

user profile. This information was used to provide services according to users preferences. Figure 4.2 shows information collected from the user, title, first name, second name etc. This information is captured in a database connected to this interface.

LogIn component - was used to identify users requesting a service, so as to be able to track user's preferences, and to provide services according to their request. Figure 4.3 shows the Log In interface.

A Service supplier component, which could be a hotel, car, and flight represent tourism service suppliers. Each of these suppliers was responsible for checking service availability, make reservation according to user's preferences and to issue a reservation confirmation once the reservation process is complete.

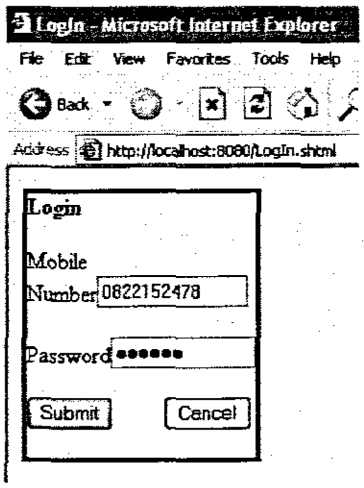

LogIn - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Address http://localhost:8080/LogIn.shtml
Login
Mobile Number 0822152478
Password ••••••
Submit
Cancel

Figure 4.3 Log-in interface

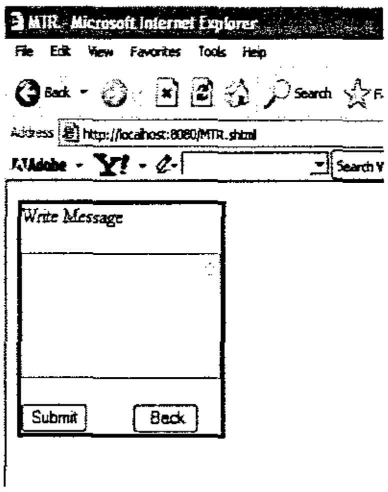

MTR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Address http://localhost:8080/MTR.shtml
Write Message
Submit
Back

Figure 4.4 Mobile travel reservation application interface

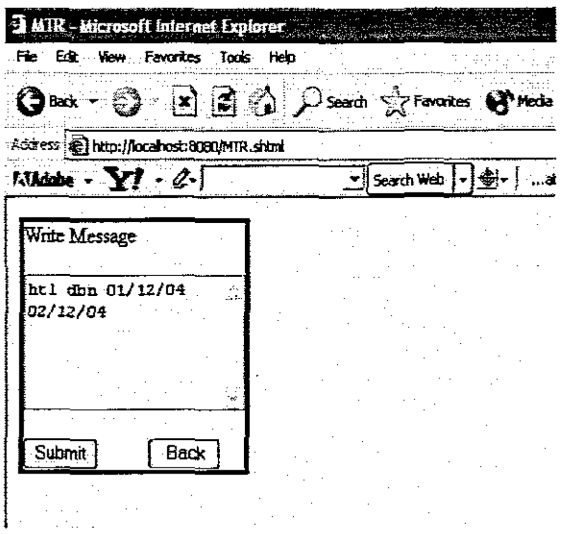
Payments component- This component was used to handle a user's payments. Most payments are done using a credit card. The Payments were used to store credit card information.

Service Requester component - this component represents the client who is interacting with the application. This class was designed to hold the client attributes.

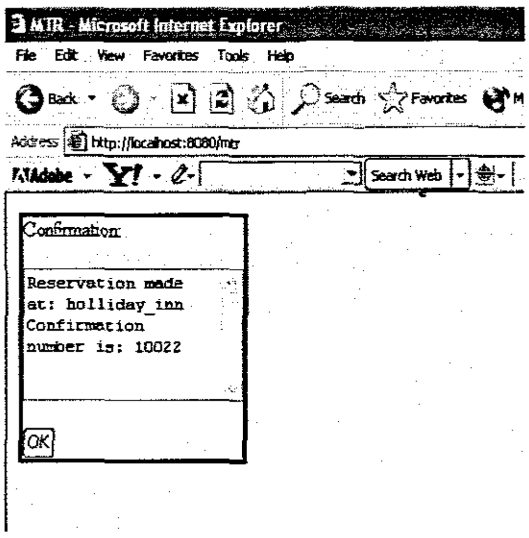
Mobile Travel Reservation (MTR) component provides access to all services provided by the application. The mobile travel reservation interface is shown in figure 4.4. It consists of message text box, the 'back' button and the 'submit' button. The message textbox was used to write a message request. The 'back' button was used to exit the application and the 'submit' button to send the message request. The next section discusses how an enterprise component such as hotel, flight and car services can be used in mobile devices.

4.2.1 Hotel Reservation Service

The message shown in figure 4.5 requests a hotel reservation accommodation service in Durban (dbn) and specifies the check-in-date and the check-out-date. Once the requester has sent a message to the application, the reservation process takes place. If the requester has set the preferences as follows: the Holiday Inn as his first preference, Protea as the second preference and, Royal Hotel as his third preference, the application checks the first preference e.g. Holiday Inn to find if there are any rooms available. If there are, the reservation is made and the confirmation is displayed as shown in figure 4.6.


MTR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/MTR.shtml
Write Message
Submit
Back

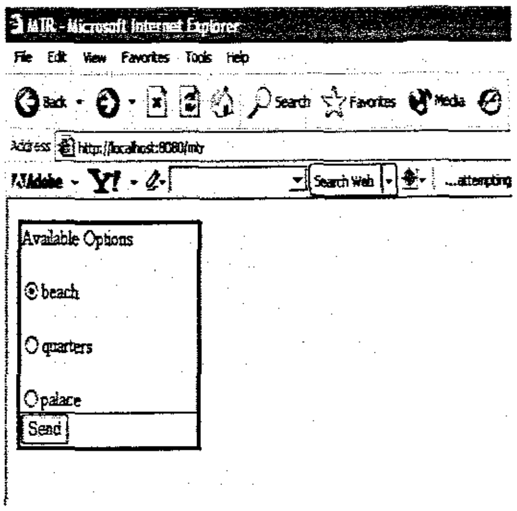
```
htl dbn 01/12/04
02/12/04
```


MTR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/mtr
Confirmation
OK

```
Reservation made
at: holliday_inn
Confirmation
number is: 10022
```

Figure 4.5 Hotel accommodation request **Figure 4.6** Hotel reservation confirmation

If there are no rooms available, the second preference is checked to find if there are available rooms. If there are rooms available the reservation is made and the confirmation is issued. If there are no rooms available, the third preference is also checked. If there are no rooms available the choice prompt (shown in figure 4.7) is displayed to show other hotels that have available accommodation service. The requester selects one hotel and the reservation takes place and the confirmation is displayed as shown in figure 4.8.

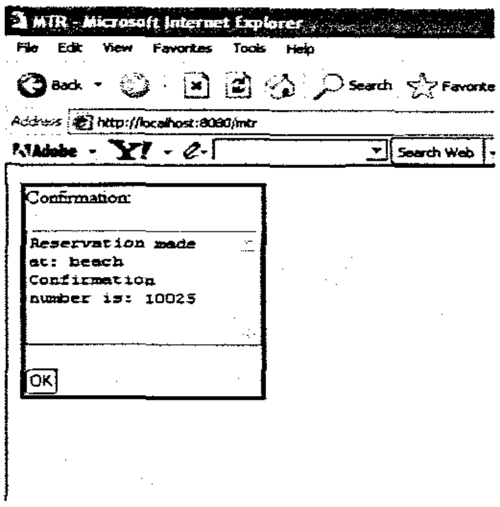

MTR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/mtr
Send

Available Options

● beach

○ quarters

○ palace


MTR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/mtr
Confirmation
OK

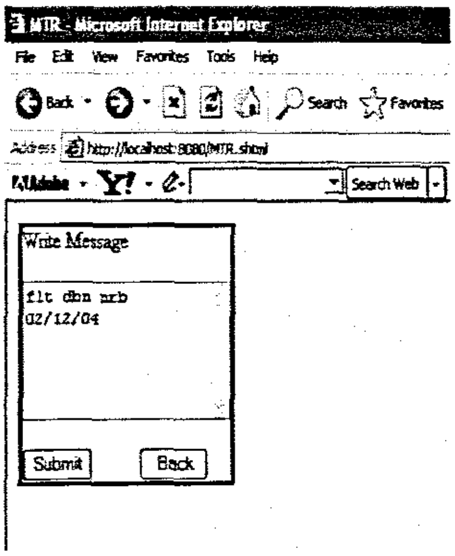
```
Reservation made
at: beach
Confirmation
number is: 10025
```

Figure 4.7 hotels choice prompt **Figure 4.8** hotel reservation confirmation

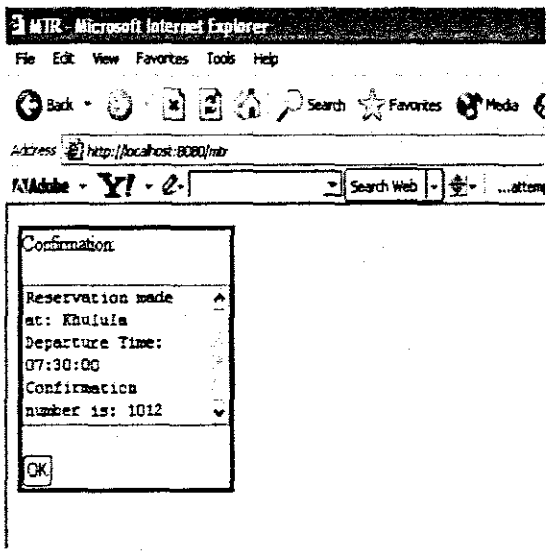
If the service requester has selected the Beach Hotel, a reservation is made at the Beach Hotel and the confirmation is displayed as shown in figure 4.8.

4.2.2 Flight Reservation Service

The message showed in figure 4.9 requests a flight reservation service. The requester has specified the departure city Durban (dbn), the arrival city Richards Bay (nrb), and the reservation date (02/12/04). If the requester has set the preferences as follows: Khulula Airways as his first preference, the South African Airways (SAA) as his second preference and the nationwide as his last preference. The first preference e.g. Khulula Airways is checked to find if there is any flight seat that is available. If there is an available seat, the reservation is made and the confirmation number is displayed as shown in figure 4.10.


MTR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/MTR.shtml
Search Web
Write Message
Submit
Back

```
flt dbn nrb
02/12/04
```


MTR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/mtr
Search Web
Confirmation
OK

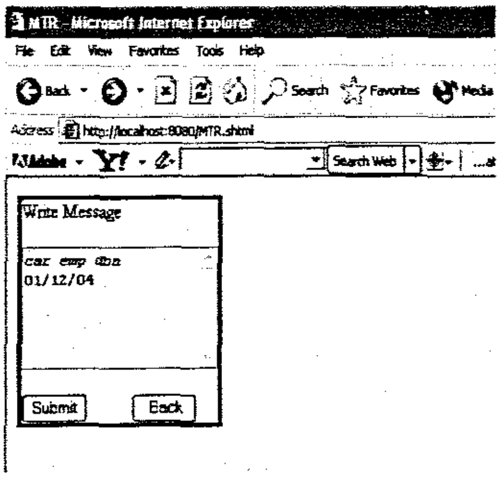
```
Reservation made
at: Khulula
Departure Time:
07:30:00
Confirmation
number is: 1012
```

Figure 4.9 Flight reservation request

Figure 4.10 Flight reservation confirmation

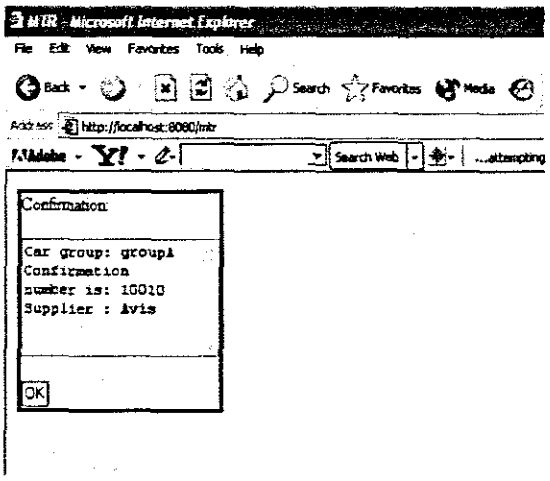
4.2.3 Car Reservation Service

The message shown in figure 4.11 shows the car reservation request. The requester specifies the pick-up-city Empangeni (emp) and the drop-off-city Durban (dbn) and the reservation date 01/12/04. If the requester has specified the preference by car groups as follows group A as his first preference and group C as his second preference and group B as his third preference. The application checks the first preference if there are any group A cars available. If there are available the reservation is made and the confirmation is displayed as shown in figure 4.12


MTR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/MTR.shtml
Search Web
Submit
Back

Write Message

```
car emp dbn
01/12/04
```


MTR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/mtr
Search Web
...attempting
OK

Confirmation:

```
Car group: groupA
Confirmation
number is: 10010
Supplier : Avis
```

Figure 4.11 Car reservation request **Figure 4.12** Car reservation confirmation

If there are no group B cars available, the second preference group C cars are also checked if there are available. The reservation is then made and the confirmation is issued.

The next section discusses the components cataloged in a UDDI registry.

4.3 Demonstration of the component repository

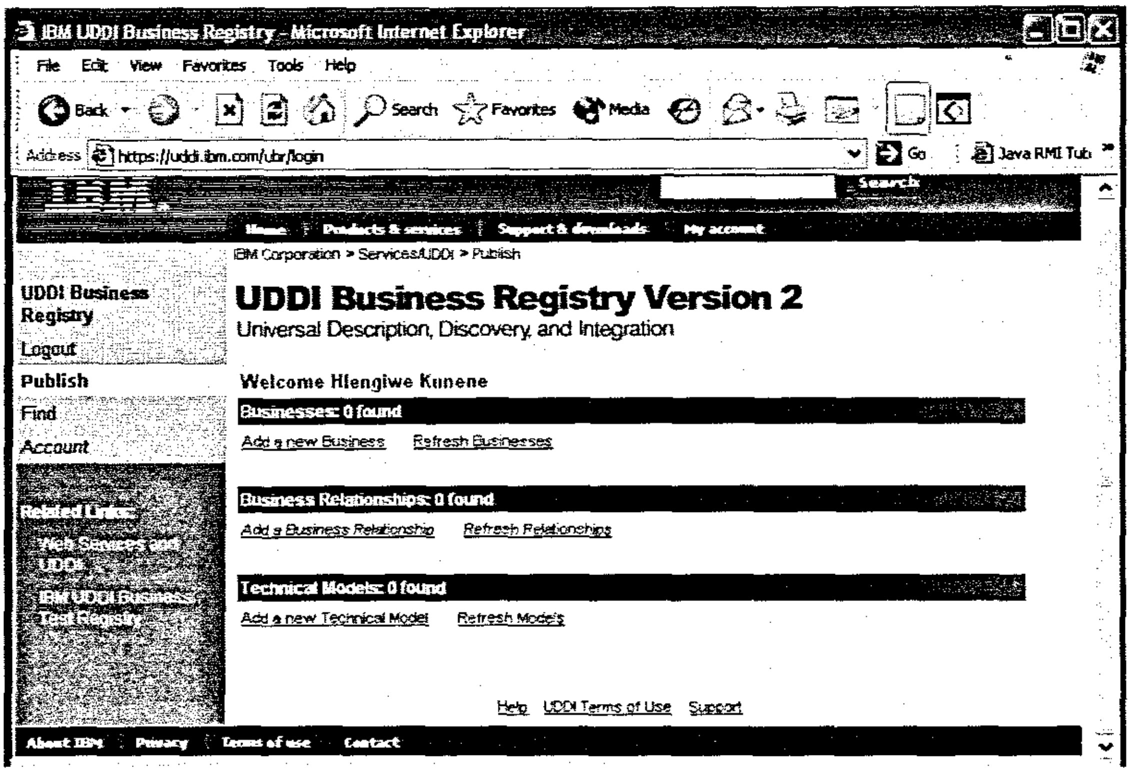
The UDDI registry was used to demonstrate eTOCOR components. The advantage of using UDDI was eliminating the complexity of components navigation, maintaining and implementing connections. In UDDI, WSDL is used to provide a description of stored entity. The UDDI provides three features, publish, find and edit which can be used to manipulate an entry in a repository. These three features were used to store, retrieve and update components respectively. Table 4.1 shows the mapping between eTOCOR features and web service technologies. The next sections demonstrate how the UDDI features were used to demonstrate eTOCOR features.

Repository Feature	UDDI counterpart feature
Store	Publish
Retrieve	Find
Update	Edit
Component name	Business name

Table 4.1 Mapping eTOCOR architecture to Web Service Technologies

4.3.1 The Store Feature

The environment of UDDI is made up of two frames: the left and the right frames as shown in figure 4.13 and all features are on the left frame. A click on the given feature (example publish) opens a link to the right frame, which contains steps to store a component. The hotel reservation component was used as an example to show how to store a component in a repository. The first step on storing a component was to activate the "Add a new Business" label which led to specification of the component name and description shown in figure 4.14. In the business table, the plus sign (+) preceding the hotel reservation component name, allowed to add services provided by the component.


IBM UDDI Business Registry - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://uddi.ibm.com/ubr/login
Go
Java RMI Tut
Search
Home Products & services Support & downloads My account
IBM Corporation > Services/UDDI > Publish
UDDI Business Registry
Logout
Publish
Find
Account
Related Links:
Web Services and UDDI
IBM UDDI Business Test Registry
Add a new Business
Refresh Businesses
Add a Business Relationship
Refresh Relationships
Add a new Technical Model
Refresh Models
Help UDDI Terms of Use Support
About IBM Privacy Terms of use Contact

UDDI Business Registry Version 2

Universal Description, Discovery and Integration

Welcome Hlengiwe Kunene

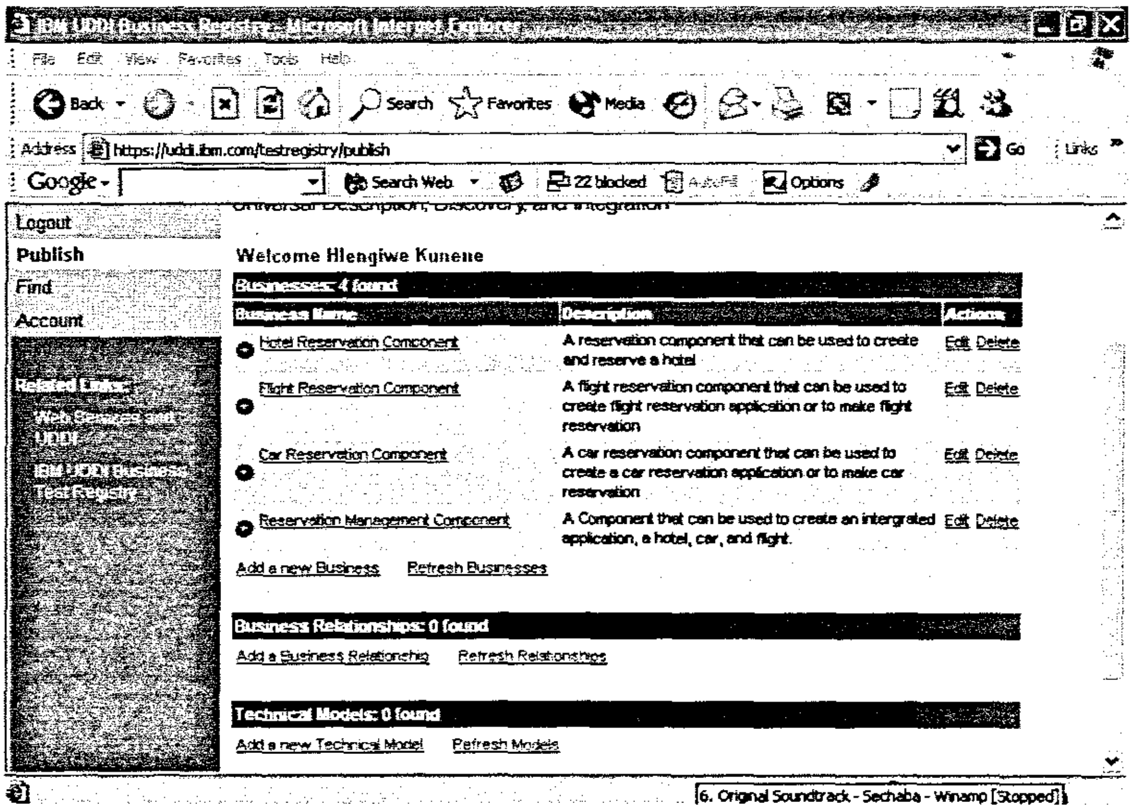
Businesses: 0 found

Business Relationships: 0 found

Technical Models: 0 found

Figure 4.13 UDDI environment

A service is specified by a name, description, and the access point. The access point of a service is specified by a URL, which points to the service itself. The next step is optional and has the aim of adding a component relationship if there is any. The last step was to add a technical model of the component. The technical model is characterized by name, the description, and overview URL. The technical model in UDDI is used to categorize all entities stored in a repository. It does this automatically according to the technical description of a component and service provided. In this case the "hotel reservation component" name was given as a technical model; this name was used as search keywords for retrieving a component, the URL was then given to show the interface of the component. After storing the new component into the component repository, it could then be retrieved anytime.



Welcome Hlengiwe Kunene

Businesses: 4 found

Business Name	Description	Actions
Hotel Reservation Component	A reservation component that can be used to create and reserve a hotel	Edit Delete
Flight Reservation Component	A flight reservation component that can be used to create flight reservation application or to make flight reservation	Edit Delete
Car Reservation Component	A car reservation component that can be used to create a car reservation application or to make car reservation	Edit Delete
Reservation Management Component	A Component that can be used to create an intergrated application, a hotel, car, and flight.	Edit Delete

Add a new Business   Refresh Businesses

Business Relationships: 0 found

Add a Business Relationship   Refresh Relationships

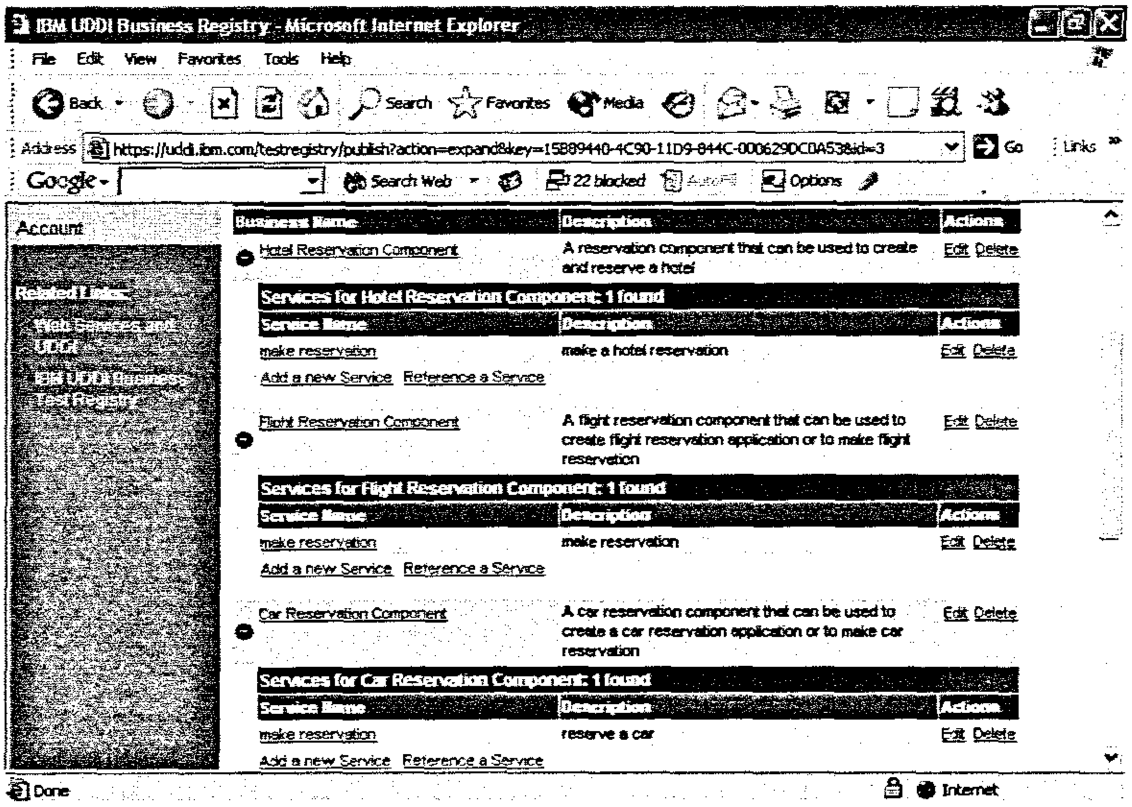
Technical Models: 0 found

Add a new Technical Model   Refresh Models

IBM UDDI Business Registry - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://uddi.ibm.com/testregistry/publish
Go Links
Google Search Web 22 blocked AutoFill Options
Logout
Publish
Find
Account
Related Links:
Web Services and UDDI
IBM UDDI Business Test Registry
6. Original Soundtrack - Sechaba - Winamp [Stopped]

Figure 4.14 Published Components

Underneath the Business Name shown in figure 4.14 is an indication of the stored components, and the plus sign shown in the left, provides services provided by a component.


IBM UDDI Business Registry - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://uddi.ibm.com/testregistry/publish?action=expand&key=15889440-4C90-11D9-844C-0006290C0A53&id=3 Go Links
Google Search Web 22 blocked AutoFill Options
Account
Related Links:
Web Services and UDDI
IBM UDDI Business Test Registry
Done
Internet

Business Name	Description	Actions
Hotel Reservation Component	A reservation component that can be used to create and reserve a hotel	Edit Delete

Services for Hotel Reservation Component: 1 found

Service Name	Description	Actions
make reservation	make a hotel reservation	Edit Delete

Add a new Service Reference a Service

Business Name	Description	Actions
Flight Reservation Component	A flight reservation component that can be used to create flight reservation application or to make flight reservation	Edit Delete

Services for Flight Reservation Component: 1 found

Service Name	Description	Actions
make reservation	make reservation	Edit Delete

Add a new Service Reference a Service

Business Name	Description	Actions
Car Reservation Component	A car reservation component that can be used to create a car reservation application or to make car reservation	Edit Delete

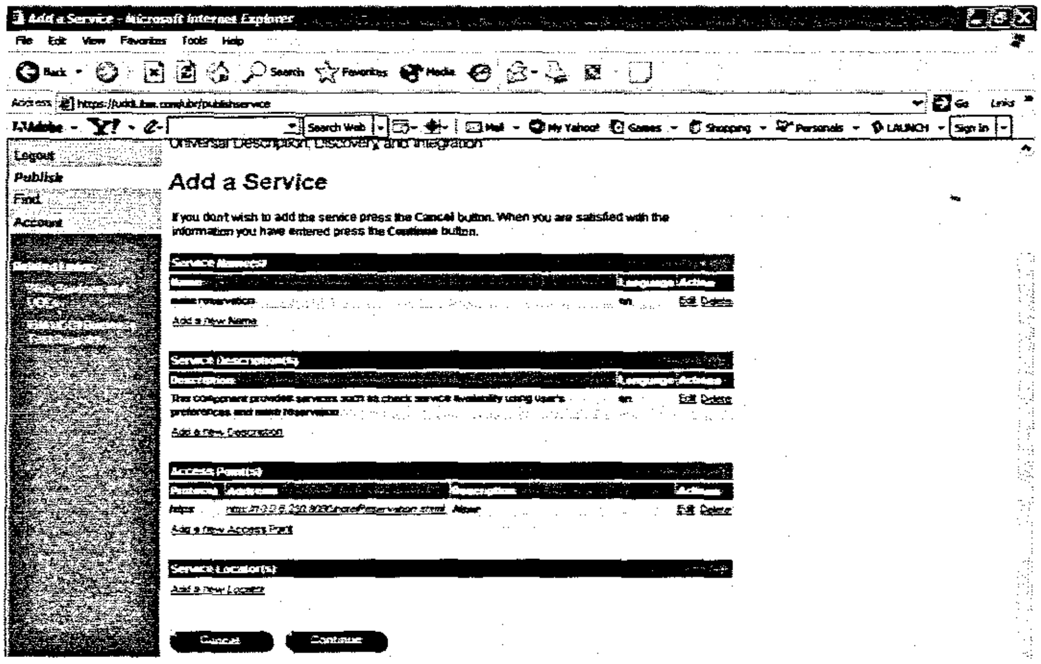
Services for Car Reservation Component: 1 found

Service Name	Description	Actions
make reservation	reserve a car	Edit Delete

Add a new Service Reference a Service

Figure 4.15 Components and services

Figure 4.15 shows components and associated services. A new service can be added by activating the "Add a new Service" label. Figure 4.16 shows the required service properties, when storing a component. Underneath the service table, the name of a service was given, and its description. The access point was also given which points to the service. Each service is also associated with a technical model. The technical model has a name, description, and the URL.



Add a Service

If you don't wish to add the service press the Cancel button. When you are satisfied with the information you have entered press the Continue button.

Service Name(s)

Name	Language	Actions
hotel reservation	en	Edit Delete

Add a New Name

Service Description(s)

Description	Language	Actions
This component provides services such as check service availability using user's preferences and make Reservation	en	Edit Delete

Add a New Description

Access Point(s)

Protocol	Address	Description	Actions
https	[illegible]/hotelReservation.shtml	None	Edit Delete

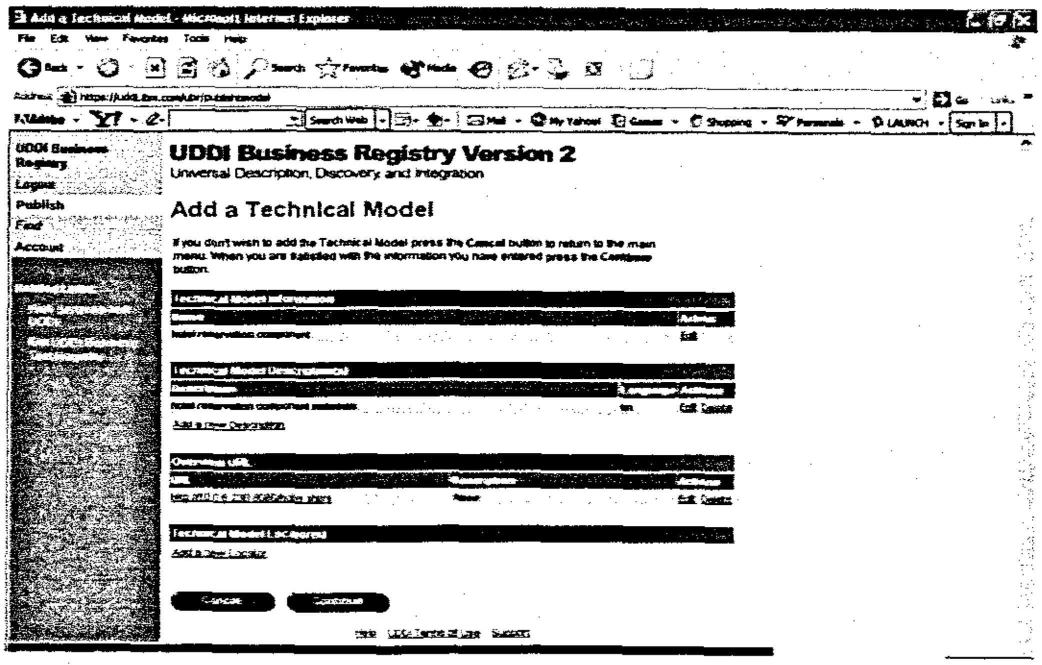
Add a New Access Point

Service Locator(s)

Add a New Locator

Figure 4.16 Service properties

Figure 4.16 shows the required properties of the service and the hotel reservation component was used to demonstrate the service properties.



UDDI Business Registry Version 2

Universal Description, Discovery and Integration

Add a Technical Model

If you don't wish to add the Technical Model press the Cancel button to return to the main menu. When you are satisfied with the information you have entered press the Continue button.

Technical Model Information

Name	Actions
hotel reservation component	Edit

Technical Model Description(s)

Description	Language	Actions
hotel reservation component [illegible]	en	Edit Delete

Add a New Description

Overview URL

URL	Description	Actions
[illegible]	None	Edit Delete

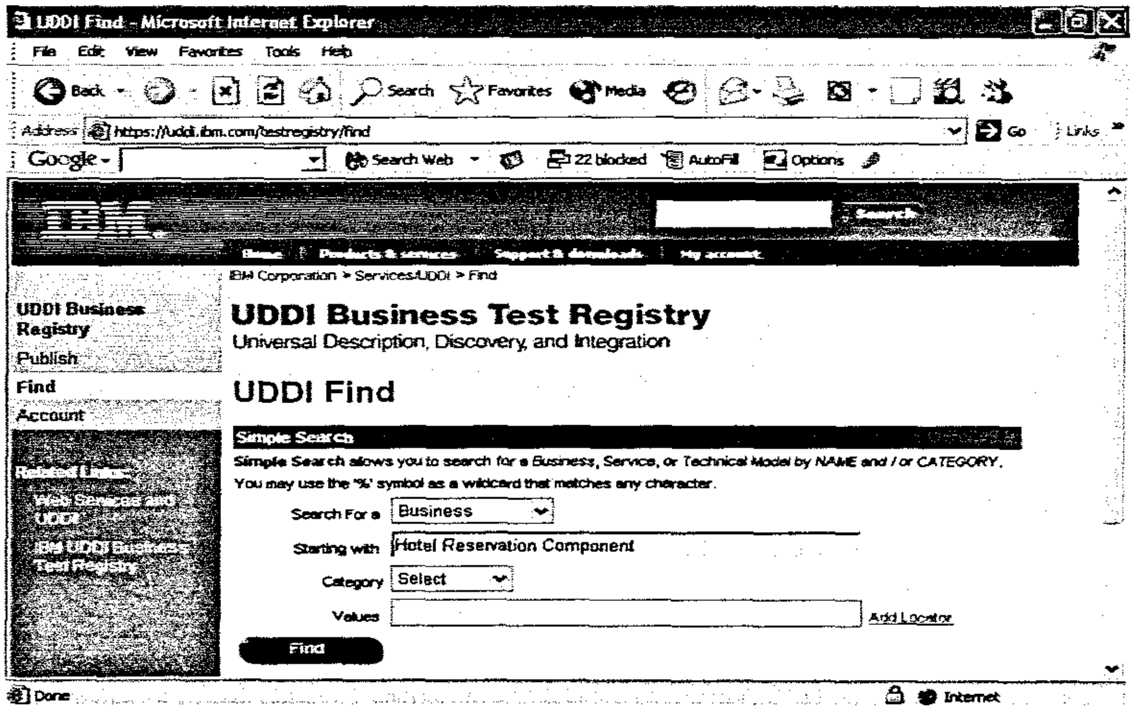
Technical Model Locator(s)

Add a New Locator

Figure 4.17 Hotel reservation technical model.

4.3.2 The Retrieve Feature

To retrieve a stored component in a repository, the UDDI find feature on the left frame of the UDDI environment was used.


UDDI Find - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://uddi.ibm.com/testregistry/find
Go Links
Google Search Web 22 blocked AutoFill Options
IBM Corporation > Services/UDDI > Find
UDDI Business Registry
Publish
Find
Account
Related Links
Web Services and UDDI
IBM UDDI Business Test Registry
Add Locator
Find
Done
Internet

UDDI Business Test Registry

Universal Description, Discovery, and Integration

UDDI Find

Simple Search

Simple Search allows you to search for a Business, Service, or Technical Model by NAME and / or CATEGORY. You may use the '%' symbol as a wildcard that matches any character.

Search For a: Business

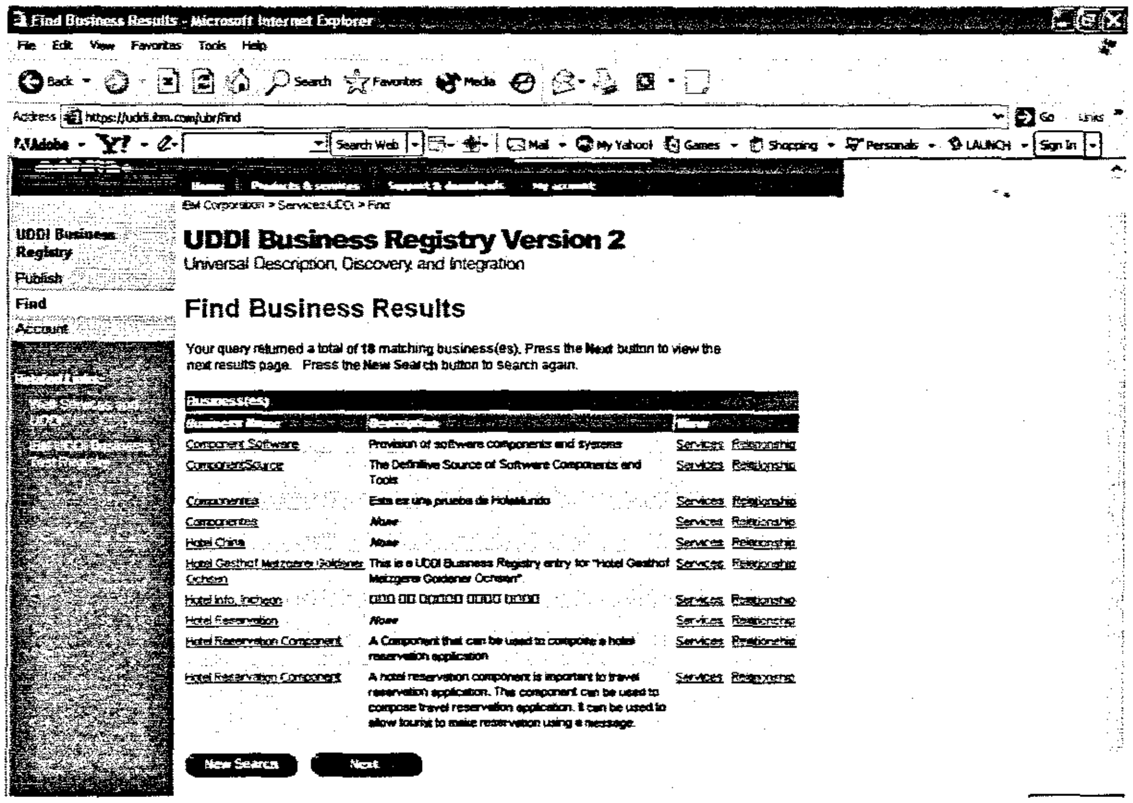
Starting with: Hotel Reservation Component

Category: Select

Values:

Figure 4.18 The search interface of the repository

The UDDI's find link that appeared on the right frame has four dialog boxes. The first "search For a" combo box was used to specify either the search is on a component or a service or a technical model. The "Starting with" textbox was used to enter the keyword to specify the search. The other two search boxes are optional, for example to retrieve a hotel reservation component stored previously (section 4.3.1) on the search for combo box, the business option was selected and a hotel reservation component was written on the "starting with" textbox. Any hotel, reservation or component keyword leads to the same result. After all the information is completed the find button which is triggered, and the results are displayed on a table as shown in figure 4.19.


Find Business Results - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address https://uddi.ibm.com/ubr/find
Search Web Mail My Yahoo! Games Shopping Personals LAUNCH Sign In
Home Products & services Support & downloads My account
IBM Corporation > Services/UDDI > Find
UDDI Business Registry
Publish
Find
Account
New Search
Next

UDDI Business Registry Version 2

Universal Description, Discovery and Integration

Find Business Results

Your query returned a total of 18 matching business(es). Press the **Next** button to view the next results page. Press the **New Search** button to search again.

Business(es)

Business Name	Description	View	
Component Software	Provision of software components and systems	Services	Relationship
ComponentSource	The Definitive Source of Software Components and Tools	Services	Relationship
Componentes	Esta es una prueba de Hotelmundo	Services	Relationship
Componentes	*None*	Services	Relationship
Hotel China	*None*	Services	Relationship
Hotel Gasthof Metzgerei Goldener Ochsen	This is a UDDI Business Registry entry for "Hotel Gasthof Metzgerei Goldener Ochsen".	Services	Relationship
Hotel Info, Incheon	[illegible]	Services	Relationship
Hotel Reservation	*None*	Services	Relationship
Hotel Reservation Component	A Component that can be used to compose a hotel reservation application	Services	Relationship
Hotel Reservation Component	A hotel reservation component is important to travel reservation application. This component can be used to compose travel reservation application. It can be used to allow tourist to make reservation using a message.	Services	Relationship

Figure 4.19 A Typical Search result

On the table of the result of each component is a name that has a service link that can be viewed and used.

4.3.3 The Update Feature

To update a component is to make changes to the component specification e.g. component name, description, service name, etc. This action is made possible by choosing once again the publish feature, in the UDDI environment. The update could be made on both businesses and technical table, the edit actions restart the process of storing information. The delete action completely removes the component or service in the repository.

4.4 Prototype Limitations

i. Service Requesters are required to know the abbreviation used to request a service;

ii. Service Requesters are required to know the format of a message and

iii. If the system is extended, it will require the Service Requester to update their preference on the website and to know the new message format of a newly-added service.

4.5 Comparison with existing IRE

The existing IRE prototype uses choice prompt to elicit specific requirements or to elicit a request from the Service Requester. This approach is interactive, where services are provided in a hierarchy. The request has to be made specific by the IRE component. It generates choice prompt for service requester, allowing him/her to specify his/her service requirements. When the requirements are specific enough, the relevant service is finally returned to the requester. The disadvantages of the existing prototype are that, the process of request/response between the requester and the system takes a considerable amount of time before the service is delivered. The IRE-enabled mobile travel reservation application uses a text message to elicit a request. The message includes the hierarchy in one screen; it summarizes the levels of screen used in the existing IRE system. The service requester does not need to wait for a response to be returned; instead a message is received in response to the request and this can be viewed later.

CHAPTER FIVE

5.0 CONCLUSION

5.1 Conclusion

An architectural mechanism that raises awareness about the uniqueness of mobile commerce when compared to electronic commerce has been presented in this dissertation. Arising from Information Requirement Elicitation approach and the home-based Mobile Commerce Reference Architecture, this research demonstrated that a repository of services is required to make service delivery to mobile commerce end-users as friendly as possible. In order to achieve the foregoing, the first result that the research produced is a comparison apparatus for showing that m-commerce is different from e-commerce.

The uniqueness of m-commerce is the raison-dêtre for a reference architecture which was earlier crafted in the Department. The repository architecture proposed in this work contributes to the usability of the existing reference architecture. The repository architecture is expected to become the core component underlying the technology archetype in the reference architecture. In order to demonstrate the repository architecture the UDDI Infrastructure has been used. However, components are needed to demonstrate how the repository works. The implementation of the mobile travel reservation application yielded the required test components.

There are some limitations of this repository architecture as a mechanism for demonstrating the role of enterprise components in future rendering of mobile commerce services .

These include:

i. Components used in this work which are not standard enterprise components;
ii. The repository architecture was not built but rather an existing counterpart, UDDI was used;
iii. The performance characteristic of the repository has not been studied and
iv. IRE could not be demonstrated as an operating environment but only as a protocol for service delivery in the application.

5.2 Future Work

It is envisaged that most of the limitations tested in the previous section will be addressed in future research work. It is of outmost importance, that the repository architecture is prototyped. Then a performance characteristic study can be conducted. It is only then that the repository architecture can be compared empirically with similar initiatives. Further more, proper enterprise components, perhaps from COTS (Commercial-off-the -shelf) software, will be used in future to test the usability of the repository.

References

[1] http://www.idt.mdh.se/cbse-book/presentations/03-chapterWC.ppt, Building Reliable Component based Systems.

[2] Witt, B. Spider Architecture Series, http://www.spiderlogic.com/news_threads/articles/SpiderArchitectureSeries_SOA.html.

[3] Schmitz et al, "Publishing your services:UDDI", http://www.ibm.com/developerWorks.

[4] Fingar, P. "Component-Based Frameworks for E-Commerce",CACM, p61-66, 2000.

[5] Adigun, M.O. "Software Infrastructure for E-commerce and E-Business Working paper." Res-CSD-01, Centre for Mobile e-Services, University of Zululand, 2004, 22p.

[6] Sun, J. Peter In, H. and Aji Sukasdadi, K. "A Prototype of Information Requirement Elicitation in m-Commerce", Proceedings of the IEEE International Conference on E-Commerce(CEO'03).

[7] Sun, J. "Information Requirement Elicitation in Mobile Commerce", CACM46(12), p 45-47.

[8] Crnkovic, I. and Larsson, M. "Component-Based Software Engineering – New Paradigm of Software Development" www.mrtc.mdh.se/publications/0293.pdf.

[9] Crnkovic, I. Hnich, B. Jonsson, T. and Kizltan, Z. "Specification, Implementation and Deployment of Components", CACM 45(10), pp 35-40, 2002.

[10] http://www.objecttool.com.

[11] http://www.whatis.com.

[12] Young, E.L. and Benbasat, I. "Interface Design for M-Commerce", CACM 46(12), pp 49-52, 2003.

[13] Diamelle Technologies, "Enterprise Business Components", www.diamelletechnologies.com.

[14] Johnson, V. and Rubin, B. "The San Francisco Project: Business Process Components and Infrastructure", ACM, p25-29, 2000.

[15] Perrey, R. and Lycett, M. "Service-Oriented Architecture", Proceedings of the 2003 Symposium on Applications and the Internet Workshops SAINT'03.

[16] Baresi, L. Heckel, R. Thone, S. and Varro, D. " Modeling and Validation of Service-Oriented Architectures:Application vs. Style ", Software Engineering Notes 28(5) pp 68-76, 2003.

[17] Papazoglou, M.P. and Georgakopoulos, D. "Service Oriented Computing", CACM 46(10), pp 25-28,2003.

[18] Yang, J. "Web Service Componentization", CACM 46(10), pp 35-40, 2003.

[19] Thompson, T. Weil, R. and Wood, M.D. "CPXe: Web Services for Internet Imaging", IEEE Computer Society, 2003.

[20] Chakraborty, D. and Chen H, "Service Discovery in the Future for Mobile Commerce",http://www.acm.org/crossroads/xrds7-2/service.html.

[21] Bass, L. Clement, P. and Kazman, R. Software Architecture in Practice, Addison

Wisley, 1998.

[22] Crnkovic, I. Hnich, B. Jonsson, T. and Kiziltan, Z. "Specification, Implementation, and Deployment of Components", CACM, p35-40.

[23] Grishikashvili, E. Reilly, D. Badr, N. and Taleb-Bendiab, A. "From Component-Based to Service-Based Distributed Applications Assembly and Management", Proceedings of the 29th EUROMICRO Conference "New Waves in System Architecture(EUROMICRO)", IEEE Computer Society, 2003.

[24] Deri, L. "A Component-Based Architecture for Open, Independently Extensible Distributed Systems", https://ramwww.unibe.ch /~deri, 1997.

[25] Arsanjani, A. "Developing and Integrating Enterprise Components and Services", CACM, 45(10),p 31- 34, 2002.

[26] Zimmermann, O. Krogdahl, and P. Gee, C. "Elements of Service-Oriented Analysis and Design", http://www-106.ibm.com/developerworks/ library/ws-soad1/, 2004.

[27] Endrei, M. Ang, J. Arsanjani, A. Chua, S. Comte, P. Krogdahl, P. Luo, M. and Newling, T. "Patterns: Service-Oriented Architecture and Web Services", http:www.ibm.com/redbooks.

[28] Katz, S. Glossary of Software Reuse Terms. Gaithersburg, MD: National Institute of Standards and Technology, 1994.

[29] Scacchi, W. "Enterprise System Analysis:Specification and Modeling", http://www.ics.uci.edu/~wscacchi/SA/Analysis/Concepts/Notes.html, Spring 2003.

[30] Bruegge, B. and Dutoit, A. Object-Oriented Software Engineering, Conquering Complex and Changing Systems, Prentice Hall, 2000.

[31] Mekonnen, G. "Information technology: It's Uses in Tourism Industry" www.ethiopiaknowledge.org/Final%20Papers/IT%20in%20Tourism, %20Mekonnen.pdf.

[32] http://www.uml.org.

[33] Levi, K. and Arsanjani, A. "A Goal-driven Approach to Enterprise Component Identification and Specification,"CACM, pp 45-52, 2002.

[34] Arsanjani, A. "A Domain-Language Approach to Designing Dynamic Enterprise Component-based Architectures to Support Business Services", IEEE Computer Society, 2001.

[35] Gamma, E. Helm, R. Johnson R, and Vlissides S." Design Patterns, Elements of reusable object-oriented Software", Addison Wesley Professional Computing Series, 1994.

[36] Moertiyoso, N. Choong Yow, K. Designing Wireless Enterprise applications on Mobile Devices, ICTA2002 ISBN: 1-86467-114-9.

[37] http://www.java.sun.com.

Appendix A : Snippets for a Hotel Reservation Component

Listing 1 : Snippets for connecting to a database

```
public void connectToDB( String DRIVER, String url, Connection con )
 {
   try {

     Class.forName( DRIVER );
     con = DriverManager.getConnection( url );
     System.out.println( "Database Connected!" );
   }

   catch( Exception e )
   {
     System.out.println( "Error occured while trying to connect to database!" );
     System.out.println( "Details" + e );
   }
 }
```

Listing 2 : Verify user id and password

```
if( request.getParameter( "login" ) != null )
  {
    String user = request.getParameter( "mobileno" );
    String password = request.getParameter( "password" );

    try {
      connection = DriverManager.getConnection( url );
      stmt = connection.createStatement();
      String query = "SELECT * FROM CRM WHERE mobile_number = '" + user + "' AND password =
      '" + password + "'";

      ResultSet rs = stmt.executeQuery( query );

      if( rs.next() )
      {
        requester.mobileNumber = user;

        if( isCardValid( requester, today, out ) )
         response.sendRedirect( "http://localhost:8080/ReservationManagement.shtml" );

        else Alert( "Your credit card has expired", "Alert", out );
      }

      else if( !rs.next() )
       Alert( "Mobile Number or Password wrong!", "Alert:", out );
    }
    catch( Exception e )
    {       out.println( "An error occured while trying to process login" );}
```

Listing 3 : The received and tokenized message

```
requestMsg = request.getParameter( "message" );

      StringTokenizer tokens = new StringTokenizer( requestMsg );

      while( tokens.hasMoreElements() )
       {
        keyword = tokens.nextToken();//pkg
        destination = tokens.nextToken();//dcitty
        adate = tokens.nextToken();//acity
        ddate = tokens.nextToken();//ddate
       }
```

Listing 4 : Snippet for Get Service Requester's Preference

```
Protected String getPreferences( ServiceRequester requester, PrintWriter out )
  {
   String preferences = "";
   try {

      Statement stmt2;

      connection = DriverManager.getConnection( url );
      stmt2 = connection.createStatement();
      String selectPrefs = "SELECT * FROM CRM WHERE mobile_number = '" +
requester.mobileNumber + "'";

      ResultSet rs2 = stmt2.executeQuery( selectPrefs );

      if( rs2.next() )
      {
        preferences += rs2.getString( "Hotel1" );
        preferences += " " + rs2.getString( "Hotel2" );
        preferences += " " + rs2.getString( "Hotel3" );
      }
   }

   catch( Exception e )
   {
      out.println( "<p>An Error occured while trying to get prefences</p>" );
      out.println( "<p>Details: </p>" + e );
      e.printStackTrace();
   }

   return preferences;
  }
```

Listing 5 : Snippet for checking service availability using service requester's preference

```
protected boolean checkServiceAvailability( String adate, String pref, PrintWriter out )
 {
  boolean isAvailable = false;

  try {

    Statement stmt3;
    int rooms = 0;

    connection = DriverManager.getConnection( url );
    stmt3 = connection.createStatement();
    String check = "SELECT * FROM Suppliers WHERE date = '" + adate + "'";

    ResultSet rs3 = stmt3.executeQuery( check );

    if( rs3.next() )
    {
      rooms = Integer.parseInt( rs3.getString( pref ) );

      if( rooms > 0 )
      {
       leftRooms = rooms - 1;
       isAvailable = true;
      }
      else if( rooms <= 0 )
       isAvailable = false;
    }
  }

  catch( Exception e )
  {
    out.println( "<p>An Error occured while trying to check service availability</p>" );
    out.println( "<p>Details: </p>" + e );
    e.printStackTrace();
  }

  return isAvailable;
 }
```

Listing 6 : Snippets for reserving a hotel using service requester's preference

```
protected void makeReservation( ServiceRequester requester, String supplier, String adate, String ddate, int
confirmationNo, PrintWriter out )
 {
   try {

      Statement stmt4, stmt5;
      String msg = "Reservation made at: " + supplier + " Confirmation number is: " + confirmationNo;
      connection = DriverManager.getConnection( url );
      stmt4 = connection.createStatement();
      stmt5 = connection.createStatement();

      String reserve = "UPDATE Suppliers SET " + supplier + " = " + leftRooms + " where date = '" +
adate + "'";
      String recordToDB = "INSERT INTO Reservations ( supplier_name, adate, ddate, confirmation_no,
mobile_no ) VALUES ('" + supplier + "','" + adate + "','" + ddate + "'," + confirmationNo + ",'" +
requester.mobileNumber + "')";

      int result1 = stmt4.executeUpdate( reserve );

      if( result1 == 1 )
      {
        int result2 = stmt5.executeUpdate( recordToDB );

        if( result2 == 1 )
        {
          Alert( msg, "Confirmation:", out );
        }
      }
   }

   catch( Exception e )
   {
      out.println( "<p>An Error occured while trying to make a reservation</p>" );
      out.println( "<p>Details: </p>" + e );
      e.printStackTrace();
   }
 }
```

Listing 7 : Snippet for billing a service requester

```
protected void hotelBill( ServiceRequester requester, String supplier, String city, PrintWriter out )
  {
    try {

       Statement stmt1, stmt2;
       double price;
       connection = DriverManager.getConnection( url );
       ResultSet rs;
       int result;

       stmt1 = connection.createStatement();
       stmt2 = connection.createStatement();

       String query1 = "SELECT * FROM Supplier where supplier_name = '" + supplier + "' AND city = '"
+ city + "'";
       String query2 = "";

       rs = stmt1.executeQuery( query1 );

       if( rs.next() )
       {
        price = rs.getDouble( "price" );
        requester.balance -= price;
        query2 = "UPDATE Payments SET credit_card_balance = " + requester.balance + " where
customer_id = '" + requester.mobileNumber + "'";

        result = stmt2.executeUpdate( query2 );

        if( result == 1 )
          System.out.println( "*************Bill has been charged!***************" );
       }
    }

    catch( Exception e )
    {
     out.println( "<p>An Error occured while trying to process billing</p>" );
     out.println( "<p>Details: </p>" + e );
     e.printStackTrace();
    }
  }
```

Listing 8 : Snippet for validating a credit card

```
protected boolean isCardValid( ServiceRequester requester, String today, PrintWriter out )
 {
    boolean isValid = false;
    try {
      Statement stmt3;
      connection = DriverManager.getConnection( url );
      stmt3 = connection.createStatement();
      int day = 0, month = 0, year = 0;//todays date
      int day2 = 0, month2 = 0, year2 = 0;//expiry date

      String checkCardNum = "SELECT * FROM Payments WHERE customer_id = '" +
requester.mobileNumber + "'";

      //---------Extracting day, month and year from today's date------//
      day = Integer.parseInt( today.charAt( 0 ) + "" + today.charAt( 1 ) );
      month = Integer.parseInt( today.charAt( 3 ) + "" + today.charAt( 4 ) );
      year = Integer.parseInt( today.charAt( 6 ) + "" + today.charAt( 7 ) + "" + today.charAt( 8 ) + "" +
today.charAt( 9 ) );

      ResultSet rs = stmt3.executeQuery( checkCardNum );

      if( rs.next() )
      {
        requester.expiryDate = rs.getString( "expiry_date" );
        requester.balance = rs.getDouble( "credit_card_balance" );

        //---------Extracting day, month and year from expiry date------//
        day2 = Integer.parseInt( requester.expiryDate.charAt( 0 ) + "" + requester.expiryDate.charAt( 1 ) );
        month2 = Integer.parseInt( requester.expiryDate.charAt( 3 ) + "" + requester.expiryDate.charAt( 4
) );
        year2 = Integer.parseInt( requester.expiryDate.charAt( 6 ) + "" + requester.expiryDate.charAt( 7 )
+ "" + requester.expiryDate.charAt( 8 ) + "" + requester.expiryDate.charAt( 9 ) );

        if( year2 > year )
          isValid = true;

        else if( year == year2 && month2 > month )
          isValid = true;

        else if( month == month2 && day2 > day )
          isValid = true;

        else isValid = false;
      }
    }
    catch( Exception e )
    {
     out.println( "<p>An Error occured while trying to check if card is valid</p>" );
     out.println( "<p>Details: </p>" + e );
     e.printStackTrace();
    }
    return isValid;
 }
```

Listing 9 : Snippet for generating a confirmation number

```
protected int getMaxConfNum( PrintWriter out )
 {
  int conf = 0;
  try {

     Statement stmt6;
     connection = DriverManager.getConnection( url );
     stmt6 = connection.createStatement();

     String getConfNum = "SELECT * From Reservations where confirmation_no = (" + "SELECT MAX(
confirmation_no ) FROM Reservations " + ")";

     ResultSet rs6 = stmt6.executeQuery( getConfNum );

     if( rs6.next() )
     {
        conf = rs6.getInt( "confirmation_no" );
        conf += 1;
     }

     else if( !rs6.next() )
        conf = 10000;
  }

  catch( Exception e )
  {
     out.println( "<p>An Error occured while trying to generate confirmation number</p>" );
     out.println( "<p>Details: </p>" + e );
     e.printStackTrace();
  }

  return conf;
 }
```

Listing 10 : snippet for confirming service requester's reservation

```
protected int getMaxConfNum( PrintWriter out )
 {
  int conf = 0;
  try {

    Statement stmt6;
    connection = DriverManager.getConnection( url );
    stmt6 = connection.createStatement();

    String getConfNum = "SELECT * From Reservations where confirmation_no = (" + "SELECT MAX(
confirmation_no ) FROM Reservations " + ")";

    ResultSet rs6 = stmt6.executeQuery( getConfNum );

    if( rs6.next() )
    {
      conf = rs6.getInt( "confirmation_no" );
      conf += 1;
    }

    else if( !rs6.next() )
      conf = 10000;
  }

  catch( Exception e )
  {
    out.println( "<p>An Error occured while trying to generate confirmation number</p>" );
    out.println( "<p>Details: </p>" + e );
    e.printStackTrace();
  }

  return conf;
 }
```

Listing 11 : Alert snapshot

```
protected void Alert( String msg, String msgType, PrintWriter out )
 {
  out.println( "<div align=\"center\">" );
  out.println( "<center>" );
  out.println( "<table border=\"3\" cellpadding=\"0\" cellspacing=\"0\" style=\"border-collapse: collapse\"
bordercolor=\"#111111\" width=\"14%\" id=\"AutoNumber1\" align=\"left\">" );
  out.println( "<tr>" );
  out.println( "<td width=\"100%\">" );
  out.println( "<p align=\"left\">" + msgType + "</p>" );
  out.println( "<font face=\"Garamond\" color = \"red\"><textarea rows=\"6\" name=\"message\"
cols=\"20\">" + msg + "</textarea></font>" );
  out.println( "<p><a href = http://localhost:8080/ReservationManagement.shtml><input type = \"button\"
name = \"OK\" value = \"OK\"></a></p>" );
  out.println( "</td></tr>" );
  out.println( "</table></center></div>" );  }
```

Appendix B : Snippets for a Flight Reservation Component

Listing 12 : Snippet for getting user requester's flight preference

```
protected String getPreferences2( ServiceRequester requester, PrintWriter out )
 {
  String preferences = "";
  try {

     Statement st;

     connection = DriverManager.getConnection( url );
     st = connection.createStatement();
     String selectPrefs = "SELECT * FROM CRM WHERE mobile_number = '" +
requester.mobileNumber + "'";

     ResultSet rs = st.executeQuery( selectPrefs );

     if( rs.next() )
     {
       preferences += rs.getString( "AirSupplier1" );
       preferences += " " + rs.getString( "AirSupplier2" );
       preferences += " " + rs.getString( "AirSupplier3" );
     }
  }

  catch( Exception e )
  {
     out.println( "<p>An Error occured while trying to get prefences</p>" );
     out.println( "<p>Details: </p>" + e );
     e.printStackTrace();
  }

  return preferences;
 }
```

Listing 13 : Snippet for checking flights availability using service requester's preference

```
protected boolean checkServiceAvailability3( Flight flight, String adate, String dcity, String acity, String
pref, PrintWriter out )
 {
  boolean isAvailable = false;

  try {

    Statement st2;

    connection = DriverManager.getConnection( url );
    st2 = connection.createStatement();
    String check = "SELECT * FROM Flights WHERE d_date = '" + adate + "' AND d_city = '" + dcity +
"' AND a_city = '" + acity + "' AND supplier_id = '" + pref + "'";

    ResultSet rs2 = st2.executeQuery( check );
    boolean found = rs2.next();

    if( found )
    {
      flight.flightCapacity  = rs2.getInt( "flight_capacity" );
      flight.flightCapacity -= 1;
      flight.flightPrice = rs2.getDouble( "price" );
      flight.depatureTime = rs2.getString( "d_time" );
      isAvailable = true;
    }
  }

  catch( Exception e )
  {
    out.println( "<p>An Error occured while trying to check service availability</p>" );
    out.println( "<p>Details: </p>" + e );
    e.printStackTrace();
  }

  return isAvailable;
 }
```

Listing 14: Snippet for reserving a flight

```
protected void makeReservation2( Flight flight, ServiceRequester requester, String supplier, String dcity,
String acity, String ddate, int confirmationNo, PrintWriter out )
 {
   try {

      Statement st3, st4;
      String msg = "Reservation made at: " + supplier + " Departure Time: " + flight.depatureTime + "
Confirmation number is: " + confirmationNo;
      connection = DriverManager.getConnection( url );
      st3 = connection.createStatement();
      st4 = connection.createStatement();

      String reserve = "UPDATE Flights SET flight_capacity = " + flight.flightCapacity + " where d_date
= '" + ddate + "' AND d_city = '" + dcity + "' AND a_city = '" + acity + "'";
      String recordToDB = "INSERT INTO AirReservations ( customer, reservation_date, confirmationNo
) VALUES ('" + requester.mobileNumber + "','" + ddate + "'," + confirmationNo + ")";

      int result1 = st3.executeUpdate( reserve );

      if( result1 == 1 )
      {
        int result2 = st4.executeUpdate( recordToDB );

        if( result2 == 1 )
        {
          Alert2( msg, "Confirmation:", out );
        }
      }
   }

   catch( Exception e )
   {
      out.println( "<p>An Error occured while trying to make a reservation</p>" );
      out.println( "<p>Details: </p>" + e );
      e.printStackTrace();
   }
 }
```

Listing 15 : Billing service requester

```
protected void flightBill( ServiceRequester requester, Flight flight, String supplier, String ddate, String
dcity, String acity, PrintWriter out )
 {
   try {

      Statement st1, st2;
      double balance;
      connection = DriverManager.getConnection( url );
      ResultSet r1;
      int r2;

      String query1 = "SELECT * FROM Payments where customer_id = '" + requester.mobileNumber +
"'";
      //String query2 = "UPDATE Payments SET credit_card_balance = " + requester.balance + " where
customer_id = '" + requester.mobileNumber + "'";

      st1 = connection.createStatement();
      st2 = connection.createStatement();

      r1 = st1.executeQuery( query1 );

      if( r1.next() )
      {
        balance = r1.getDouble( "credit_card_balance" );
        balance -= flight.flightPrice;
        String query2 = "UPDATE Payments SET credit_card_balance = " + balance + " where
customer_id = '" + requester.mobileNumber + "'";

        r2 = st2.executeUpdate( query2 );

        if( r2 == 1 )
        {
         System.out.println( "*************Bill has been charged!****************" );
         System.out.println( "Total bill: " + balance );
        }
      }
   }

   catch( Exception e )
   {
    out.println( "<p>An Error occured while trying to process billing</p>" );
    out.println( "<p>Details: </p>" + e );
    e.printStackTrace();
   }
 }
```

Appendix C : Snippet for a Car Reservation Component

Listing 16 : Snippet for car reservation using service requester's preference

```
protected void makeReservation( ServiceRequester requester, Cars c, String pref, String dcity, String acity,
String ddate, int confirmationNo, PrintWriter out )
 {
   try {

      Statement st3, st4;
      String msg = "Car gruop: " + pref + " Confirmation number is: " + confirmationNo;
      connection = DriverManager.getConnection( url );
      st3 = connection.createStatement();
      st4 = connection.createStatement();

      String reserve = "UPDATE Schedule SET " + pref + " = " + c.carsLeft + " where d_date = '" + ddate
+ "'";
      String recordToDB = "INSERT INTO CarReservation VALUES ('" + requester.mobileNumber +
"','" + ddate + "','" + pref + "'," + confirmationNo + ",'" + dcity + "','" + acity + "')";

      int result1 = st3.executeUpdate( reserve );

      if( result1 == 1 )
      {
         int result2 = st4.executeUpdate( recordToDB );

         if( result2 == 1 )
         {
            Alert( msg, "Confirmation:", out );
         }
      }
   }

   catch( Exception e )
   {
      out.println( "<p>An Error occured while trying to make a reservation</p>" );
      out.println( "<p>Details: </p>" + e );
      e.printStackTrace();
   }
 }
```