Web-based E-commerce Service Provisioning using a TINA Retailer

S. Brassell

A project report submitted to the faculty of Engineering, University of the Witwatersrand, Johannesburg, in partial fulfillment of the requirements for the degree of Master of Science in Engineering

Johannesburg, July 2001

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Science in Engineering at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other university.

_____
Shaun Brassell

Signed this _____ day of _____ (year) _____

Abstract

This project examines the use of TINA concepts to provide the required service architecture layer for the Next Generation Network. In particular, it details how the current Internet-based e-commerce environment could be integrated as a service within the TINA NGN solution. A prototype service in the form of a virtual shopping mall provides the context for this development. The presented model allows for seamless integration with existing Internet e-commerce solutions whilst providing a number of additional value-added services.

Table of Contents

List of Figures

Acronyms

TINA	Telecommunications Information Network Architecture
NGN	Next Generation Network
OSM	Open Service Marketplace
WWW	World Wide Web
DPE	Distributed Processing Environment
CORBA	Common Object Request Broker Architecture
OMG	Object Management Group
IDL	Interface Definition Language
IIOP	Internet Inter-Orb Protocol
IOR	Interoperable Object Reference
HTTP	Hyper Text Transfer Protocol
API	Application Programming Interface
IN	Intelligent Network
ISP	Internet Service Provider
B2B	Business to Business
B2C	Business to Consumer
VSM	Virtual Shopping Mall
IA	Initial Agent
UA	User Agent
SUB	Subscription Management Component
SF	Service Factory
PA	Provider Agent
SSM	Service Session Manager
USM	User Session Manager
ssUAP	Service Session User Application
Ret-RP	Retailer Reference Point
3Pty	Third Party Reference Point

1 Introduction

The telecommunications environment is currently experiencing a paradigm shift from propriety and centralised circuit-switched networks to converged packet networks. The legacy model in which voice, video, and data are delivered over separate, dedicated, single-purpose networks, no longer meets the needs of today's business requirements. By migrating to a converged packet network, service providers will be able to offer, and charge for, a richer menu of enhanced services, while at the same time lowering the costs of delivering these services.

Operators have for some time now realised the importance of differentiating themselves based on the value-added services that they offer. Early voice services were associated with a particular vendor's switching equipment, and were implemented in a proprietary manner. Service Providers were limited in what they could offer as these services were expensive to develop and took a long time to roll out. Interworking between different vendors also proved to be difficult due to the proprietary nature of these services.

In an effort to remove some of these limitations, Intelligent Networks (IN) were introduced. The IN architecture anchored services and service creation to pure telephony, so services were 'voice only'. These services were generally rooted firmly in the network and not integrated with end-user applications. The circuit switched characteristics of the Public Switched Telephone Network (PSTN) placed limitations on the distributed nature of this architecture; edge devices such as basic telephone sets remained simple. IN service creation was an intensive development exercise, and creating new services was time consuming. The overlay solution it brought was never efficient or elegant, and certainly falls short of being able to introduce the new multimedia services [1].

The telecommunications environment is now experiencing a new era in service provisioning that has been fuelled by the unprecedented growth of the Internet. The Internet environment presents an open service marketplace [2] whereby anyone has the ability of designing and deploying services. The Internet has been able to combine voice,

data, and video to provide users with rewarding interactive shopping, business, and entertainment experiences.

The Internet has also sparked an interest in IP telephony, both among Internet users and public operators alike. IP telephony originally gained popularity by allowing Internet users to bypass PSTN charges, most notably on International calls. With enhancements on Quality of Service (QoS) mechanisms for prioritising, queuing, and reserving bandwidth for the delay sensitive voice traffic, packet telephony on a managed network has now become an attractive option to network operators. Using a single network to provide both voice and data services promises significant savings over traditional circuit-switched telephony, and enables a variety of new services to the customer.

In the near future, there will be a number of new 'IT' terminals that will interact with this single network. Terminals such as softphones, PDA's, videogame consoles, or home network appliances will need to operate along side the present PSTN terminals. The single network will thus require a unified service layer that will support diverse services over a converged network. The TINA Consortium (TINA-C) [3] has proposed a Service Architecture to meet the needs of this NGN service layer. The TINA architecture defines an open, object-orientated platform that allows for service provision on a Distributed Processing Environment (DPE).

This TINA architecture has evolved over the years through Requests for Proposals (RfP) and trials that have been undertaken at a number of institutions worldwide. The South African TINA trial (SATINA) was established in 1999 to evaluate and contribute to this architecture, and has involved a number of projects conducted at the University of the Witwatersrand [4]. This project details a service that has been implemented on the SATINA platform.

1.1 Problem Statement

The approach undertaken by most incumbent operators to deploy NGN solutions involves a migration strategy from their current networks to the NGN packet solution. Migration is an attractive option to these carriers due to the PSTN representing a multi-billion rand investment on which they will continue to earn revenue [5]. Initial services offered on the NGN will therefore need to operate alongside existing PSTN services, including current services offered on the Internet.

Business-to-Consumer (B2C) e-commerce within the Internet environment is one such service. E-commerce is expanding at a rapid rate providing online transaction capabilities to millions of users. Companies have invested large amounts of their resources in developing their e-commerce solutions, and will require this environment to be made available within the NGN environment.

The focus of this project is to investigate how web-based e-commerce could be offered as a service on a network that employs the TINA Service Architecture. This process includes:

- Identifying the relevant technologies that will support the provisioning of e-commerce services to users on such a network.
- Analyzing the business model implications resulting from the incorporation of current e-commerce services into a TINA environment.
- Designing and coding a prototype service that allows a user on this TINA network to interface with existing e-commerce application servers.
- Evaluating the issues and merits that arise from this TINA E-commerce integration.

1.2 Report Outline

The remainder of this report is structured as follows:

Chapter 2 presents the relevant background information. This includes an overview of the envisaged NGN, requirements for an enhanced service architecture, the TINA architecture, and the interworking of the TINA architecture with the present e-commerce environment. The chapter concludes with an overview of the SATINA platform on which the proof-of-concept service is built.

Chapter 3 presents the design and implementation of a virtual shopping mall service. This service enables users on a TINA network to interface with existing e-commerce application servers.

Chapter 4 concludes the report with a discussion of the material presented, and gives recommendations for future work.

The **Appendix** includes a listing of the IDL code, an explanation to the TINA Service Architecture computational components, introduction to CORBA, and an overview regarding the structure of the included CD.

- ***User Naming Convention***

For the purpose of this report, a distinction is made between two broad categories of end-users. *TINA Consumers* relate to end-users on a network that employs TINA concepts. *Internet users* relate to end-users on a network that employs generalised Internet technologies.

2 Background Information

This chapter presents the relevant background information. The chapter begins with an overview of the envisaged NGN, and details how this future architecture could inter-work with the current PSTN and Internet environment. The requirement for an enhanced service architecture to support this NGN is then discussed, with a viable solution in the form of the TINA Service Architecture presented. The interworking of the Internet e-commerce environment with this TINA Service Architecture is highlighted, which includes a focus on the technologies to support this process, and the modification of the business model that occurs when applying a TINA NGN solution. The chapter concludes with an overview of the SATINA trial on which the prototype proof-of-concept service is built.

2.1 Next Generation Networks

2.1.1 Future Network Architecture

An NGN is defined as a packet-based telecommunications network that employs new processing, control, management, and signalling techniques to provide all types of services, from basic narrowband voice telephony services, to advanced broadband multimedia services [6]. The NGN comprises of a number of entities that interact across a distributed processing environment using Application Programming Interfaces (API's).

Current approaches to NGN solutions are however protocol based, with entities interfacing using a multitude of different protocols. The basic entities that encompass an NGN solution comprise of a mixture of terminals, servers, call control and management functionality, and a variety of media, signalling and access gateways. A high-level overview of these entities are depicted in figure 2.1, and are detailed below:

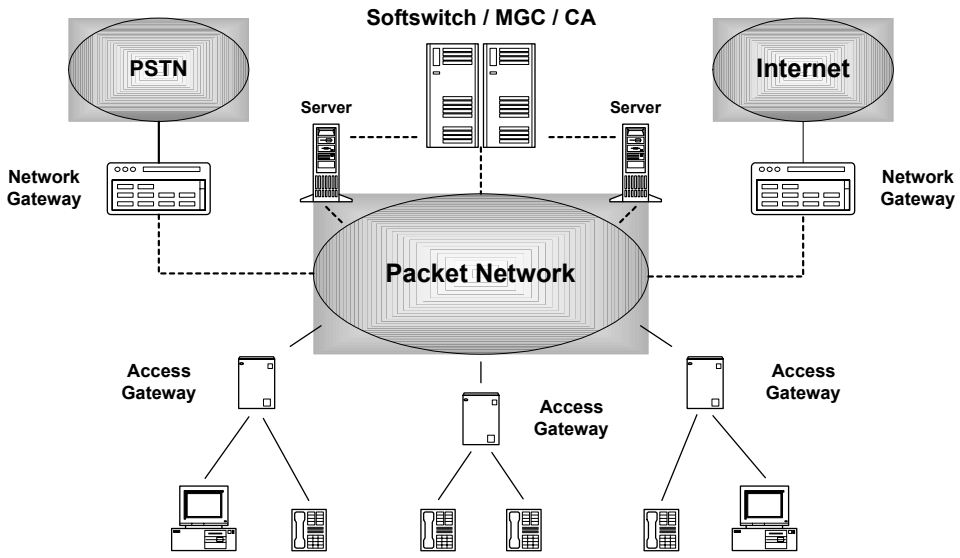

Figure 2.1: Next Generation Network

- *Terminals*

Numerous terminals supporting new generation packet based telecommunication networks exist today. These terminals include PDAs, home network appliances, and telephonic devices in the form of IP telephones, software applications for PCs (softphones), and analogue phone adapters (allowing analogue terminals to connect to an IP-based data network via a box or PC card). Telephonic terminals are usually connected to the IP network via Ethernet interfaces, and utilise numerous signalling and control protocols including the H323, SIP, H248/MEGACO or LPCP protocol [7]. The selection of protocol has a direct influence on the complexity of the terminal thus influencing the cost.

- *Servers*

There are a number of servers within the network that are used to host broadband and network services. Broadband servers comprise of application / content server farms, storage area networks (SANS), and databases. Network servers include domain name servers (DNS), authentication authorization and accounting servers (AAA), and servers for caching.

- *Gateways*

Gateways are essential components that allow the interworking between networks and entities that are logically separated, or are of differing standards. Network gateways occur between two dissimilar networks (e.g. PSTN–to-IP) and broadly comprise of signalling and media gateways.

Signalling Gateways are responsible for the termination and translation of various signalling protocols including SS7, H323, and SIP (with BICC CS2 under development [8]).

Media Gateways are responsible for the termination and translation of various media types, for example, PCM voice on E1 trunks to IP packets utilising RTP. In general media gateways consist of voice codecs, echo cancellation functionality, and tone detection and generation. There are a number of specific media gateways that have been defined which include access gateways, trunking gateways, and residential gateways. These gateways are controlled by the call control entity as detailed below.

- *Call Control / Management Functionality*

Entities to provide call control and management functionality form the heart of the NGN solution. There are a number of elements and possible solutions that provide this functionality. These include Media Gateway Controllers (MGC), H323 Gatekeepers, SIP Proxies, service layer interface entities (including API's for third party control e.g. Parlay or JAIN), and OSS interactions. These elements are packaged together in a number of different forms and are sometimes referred to as Call Feature Servers (CFS), Call Agents (CA), or Softswitches [9].

The MGC is required to assist in the interoperability of the packet network with the PSTN environment. The MGC communicates with *decomposed* media gateways using the MGCP protocol or the new H248/MegacoP protocol.

2.1.2 NGN Service Architecture Requirements

Next generation networks require a service delivery and creation platform that promotes rapid service creation and creates a basis for new value propositions [10]. The platform should be open to application development by all comers: equipment manufacturers, network operators, and third-party software developers. The platform should integrate diverse network resources and make it easy to create mixed media applications. Moreover, the new platform should stand apart from the transport and control network so that switching technology can evolve without disrupting existing services. This platform occupies the defined *service layer* portion of the NGN architecture (figure 2.2). The service layer supports programming of applications, and hides the complexity of the underlying network protocols.

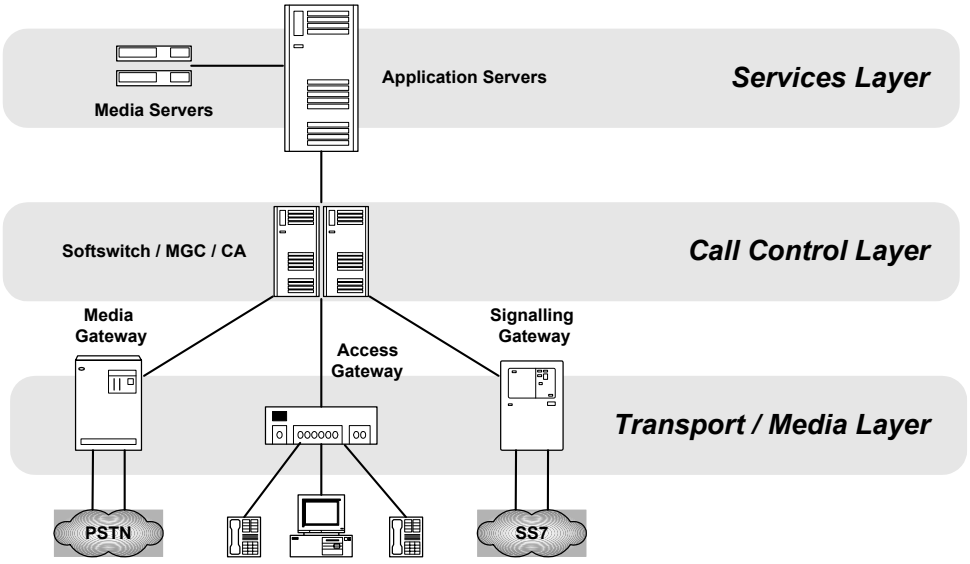

Figure 2.2: Service Layer Separation

Some of the requirements of this new service architecture include rapid service provisioning, centralised billing, service customisation and activation, user profile management, and mechanisms for authentication and security.

2.1.3 TINA Service Architecture

In an effort to meet the needs of the next generation service architecture, the TINA Consortium proposed a set of concepts and principles for the design, specification, implementation, and management of telecommunication services. The resulting TINA *Service Architecture* [11] specifies how the different roleplayers in the system (discussed in detail in section 2.2.3) interact with each other across set reference points in a system that has distributed functionality.

The basic concepts and entities that define the TINA Service Architecture include the session concept, the Service Architecture computational components, and the Distributed Processing Environment over which these components interact.

- *The Session Concept*

The TINA Service Architecture is divided up into three parts separating the access, service, and communication concerns (figure 2.3). Each part interacts within a 'session' where a session provides a context or encapsulation of related objects which might be specified over time, service, resource or role [3].

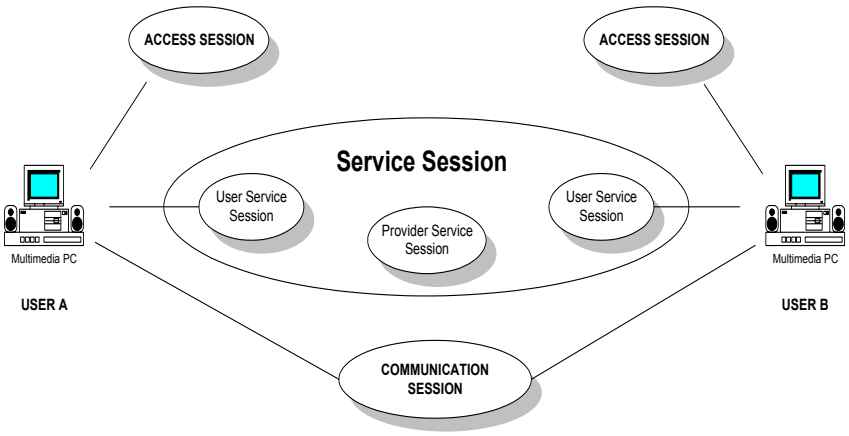

Figure 2.3 TINA Session Concepts

The ***access session*** defines and manages the user's access to the service architecture. It identifies a secure, accountable and manageable association between the user and the service provider.

The ***service session*** provides a context for one or more users to support the execution of the service whilst taking into account the equipment that the user is utilising. It details the interactions among components that are required to control the behavior and management of the service. A service session must be linked to at least one access session.

The ***communication session*** deals with the interactions that are required to establish and maintain stream (audio/video/data) connections among components implementing the service. The communication session provides a flexible mapping of service sessions onto communication sessions, and must be linked to a specific service session.

- *The Service Architecture Components*

The access, service, and communication sessions are implemented using software components (figure 2.4), with each component carrying out a certain responsibility in the provision of a service [12]. The components are divided between a *User* and *Provider* domain. These domains are generic and can apply to any two roleplayers in a TINA system (see section 2.2.3). In the interaction between a Consumer and a Retailer, the Consumer will obtain the *user* domain components whilst the Retailer contains the *provider* domain components. All interactions between these two domains are specified in a respective reference point, and interact to establish access and service sessions (see section 2.3.2).

The definitions and functionality of each component is described in Appendix B.

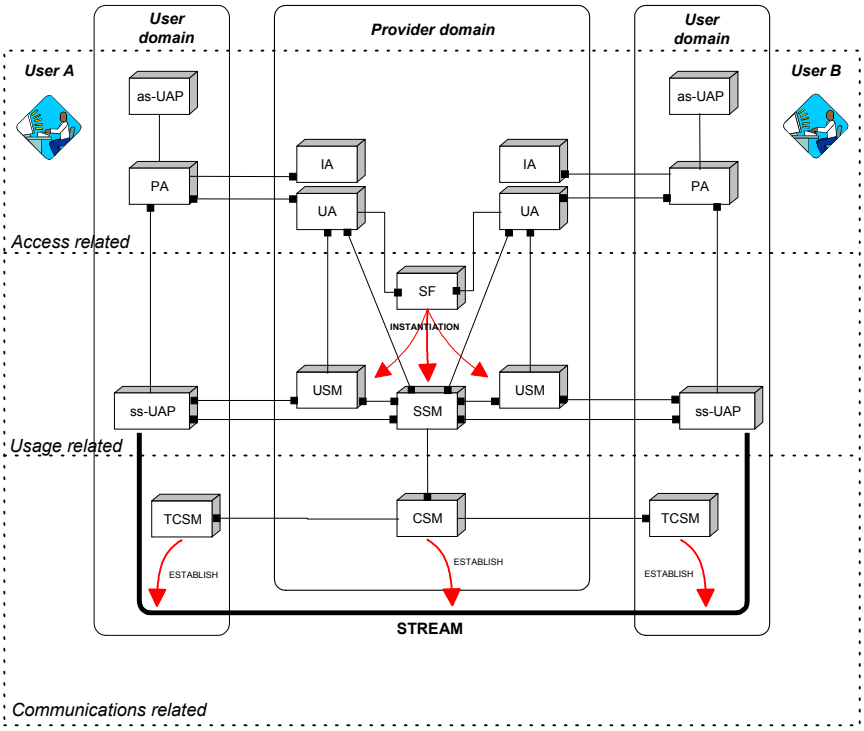

Figure 2.4 TINA Service Architecture

- *The Distributed Processing Environment (DPE)*

As discussed in the previous section, TINA specifies a number of computational components belonging to roleplayers that are distributed across the network. These components interact over a variety of distributed network platforms and programming languages.

The TINA Distributed Processing Environment (DPE) supports the distributed execution of objects, and provides a number of important features when applied to large telecommunication networks. The DPE hides the underlying characteristics of the operating system, provides location transparency, and allows for fault tolerance to the applications using it. The DPE consists of a kernel and a number of services including a naming, trading, life-cycle management, and a notification service. The DPE is not bound to any specific object middleware technology, but has adopted the OMG's CORBA architecture due to mutual work conducted by TINA-C and OMG [13].

2.2 TINA E-Commerce Service Integration

2.2.1 Providing E-Commerce Services in a TINA Environment

As operators migrate their present networks towards NGN solutions, they will require a mechanism for the interworking of the current e-commerce environment with their new service architecture. With the focus of this project on the interworking of this e-commerce environment with the TINA Service Architecture, a solution was sought that would allow the TINA Consumer access to this environment with the least amount of adaptation required on behalf of the e-commerce provider.

The designed TINA service will have to take into account the present Internet technologies that are being used to provide e-commerce services, and to allow the interworking between these technologies, and the technologies and principles proposed by TINA.

- *Background to Internet-based e-commerce*

E-Commerce first emerged in the early 1960's with the introduction of Electronic Data Interchange (EDI) between large organisations on their private networks. Although e-commerce is some thirty years old, it has mainly served the B2B (business-to-business) marketplace. With the introduction of the Internet, we have seen a surge in the B2C (business-to-consumer) model, with many businesses realising the potential of providing services over this OSM. E-commerce is not only shopping over the Internet, but encompasses every business operation that involves electronic and network technologies. B2C has evolved from product advertisement and customer feedback, through to comprehensive all-encompassing business solutions. It is envisaged that nearly $1trillion will be spent online in 2002 [14].

The most common form of B2C e-commerce on the Internet today utilises the general 'shopping basket' scenario. This scenario involves a consumer logging onto a desired site, selecting and placing the required goods in a virtual shopping basket, and purchasing

the chosen items by submitting their credit-card information. The proposed TINA service shall provide the TINA Consumer access to this specific scenario (detailed in Chapter 3). Apart from allowing the TINA Consumer access to this shopping environment, many TINA principles will be beneficial in providing solutions to some of the problems presently facing e-commerce.

- *Problems facing e-commerce*

Although the e-commerce environment is expanding at a rapid rate, users have yet to fully embrace the technology due to a number of problems that still exist [15]. Some of these problems include:

- *Security and billing issues*

  The predominant method chosen for purchasing items on the Internet involves the transfer of credit card information between the involved parties. The current support for the secure transmission of this data is however diverse, with a number of weak security schemes fuelling an increase in credit card fraud. Related to this point, there is a lack of centralised billing, and hence no singular point of payment. Users are still required to present their financial information to a number of different vendors with varying security mechanisms.

- *Lack of global user-profile management*

  With the large amount of goods becoming available for purchase on the Internet today, users are requiring mechanisms to personalise their view on this content. Different e-commerce providers allow consumers to customise their requirements, but this process is usually done in a propriety manner. There is a lack of global user-profile management, requiring the consumer to input their requirements for each provider they utilise.

- *Availability problems resulting in transaction failure*

  More increasingly, the e-commerce environment is experiencing a rise in transaction failure and hence a loss in revenue. These failures have been attributed mainly to e-commerce server failure or server overloading. To counter these problems, e-commerce providers employ costly redundant features that are sometimes out of the reach of small-scale providers.

2.2.2 E-Commerce and Object Middleware Technologies

To allow the interworking of the e-commerce environment with the TINA Service Architecture, design choices based on the underlying technologies of these two entities need to be made.

- *Internet Technologies*

Internet technologies that include the transport protocol TCP/IP, the application protocol HTTP, and the document formatting protocol HTML form the basic elements of the WWW. These elements, together with security mechanisms in the form of Secure Socket Layering (SSL), Secure HTML (SHTML), and Secure Electronic Transactions (SET) allow for basic e-commerce transactions. More advanced application servers employ a variety of technologies to dynamically create webpages, maintain user sessions, and interact with distributed backend databases. These technologies include Active Server Pages (ASP), JAVA Server Pages (JSP), cookies, Servlets, Applets, and eXtensible Markup Language (XML).

- *TINA Technologies*

TINA does not detail any specific underlying technology, but rather specifies principles and interfaces for the interaction between the objects involved. Object middleware technologies of CORBA, IDL and the Internet Inter-ORB Protocol (IIOP) form the basic elements of the TINA computing platform.

2.2.3 Internet and TINA Business Model Comparison

Delivering e-commerce services via the Internet or via the TINA framework implies a different business model for each scenario. The Internet business model stipulates a direct relationship between the user and service provider, whereas the TINA Business Model provides a 'middleman' approach to service provision. The difference between the Internet and the TINA business model is detailed below, together with the benefits that arise by utilising a TINA model.

- *The Internet Business Model*

The business model for the World Wide Web is shown in figure 2.5. Each Consumer has a relationship with an Internet Service Provider (ISP) to gain access to the Internet to use the services of the WWW for e-commerce transactions. Likewise, the Content Provider (i.e. the e-commerce provider) has a similar relationship to the ISP to gain access to the Internet to provide the relevant business content. ISP's also interact with each other to create the global Internet, but this relationship merely covers IP connectivity and does not provide a formal retailing relationship. It is also possible for consumers to have a separate direct connection to a Content Provider, and vice-versa.

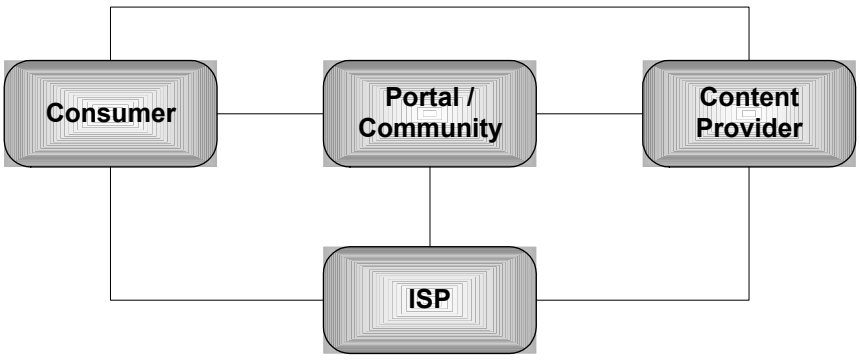

Figure 2.5 Internet Business Model

With the rapid growth of content offered over the WWW, users soon required a mechanism for personalising their content. A Portal, offering users a personal starting point for accessing the Web fulfills this need. Often the Portal brings Consumers with similar interests in contact with each other, thus creating a Community. Portals in the form of shopping malls exist for e-commerce applications offering basic services to stores and users that are affiliated to that portal.

- *The TINA Business Model*

The TINA Business Model (figure 2.6) is the enabling framework for the various business entities involved. The model separates the key business domains into different roleplayers with well-defined reference points between them. There are five main roleplayers: Retailer, Consumer, Third-Party Service Provider, Broker, and Connectivity Provider, each responsible for different aspects of service provision. Reference points have also been defined for interworking within a domain that has meaningful interactions (e.g. Ret-Ret reference point).

The most defined reference point is the Retailer Reference Point (Ret) [16] defining interactions between the Consumer and the Retailer. Various interfaces and methods have been specified across this reference point to allow the Consumer to initiate access with their Retailer, and to discover and launch a variety of services.

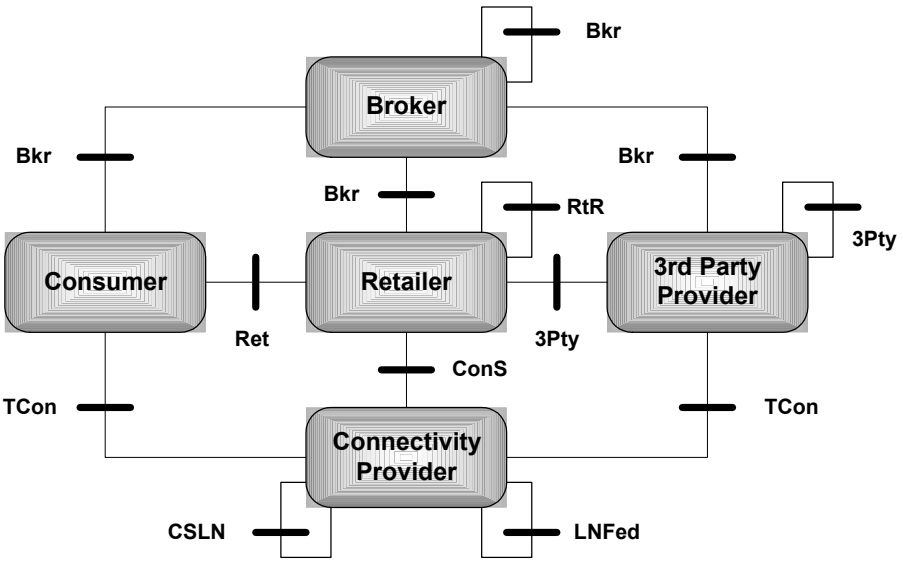

Figure 2.6 TINA Business Model

The TINA Business Model enables the Consumer to obtain many services from many Third Party Providers through a single Retailer. The Consumer only has to trust a single Retailer in order to have secure access to a wide range of services.

- *Business Model Migration Benefits*

The TINA Business Model provides several benefits to the two bodies involved in an e-commerce transaction. E-commerce providers can charge for content without having to deal with complex billing processes and customer care. Furthermore, consumers only need to have a direct formal relationship with the Retailer instead of separate relationships with all the different e-commerce providers. This means that the consumer only receives one account and only has to initiate one login procedure.

Enhanced shopping experiences are also possible due to the introduction of middleware technologies on each party. The Retailer will be able to determine the present server utilisation and capability on the e-commerce provider before directing consumers to that site for a shopping session. The Retailer is also able to maintain the Consumer's profile, which could be made available to specific e-commerce stores on request.

2.3 SATINA

2.3.1 SATINA Overview

The University of the Witwatersrand, supported by Telkom SA Limited: ITAS Division, Siemens Telecommunications, and a variety of other stakeholders have developed an experimental platform on which to evaluate the TINA architecture [4]. The SATINA trial aims to develop an experimental TINA-based services and network connectivity platform, reflecting the requirements of future telecommunication service providers and network operators.

The SATINA trial aims to:

- Foster an understanding of the convergence of telecommunications and the data network environment.
- Develop high level expertise in advanced telecommunications service architectures, service management, and network management based on a TINA Distributed Processing Environment using CORBA.
- Promote research in network and service modeling, performance, optimisation, and control.
- Develop reusable and service-specific software to allow for rapid service provisioning.
- And to assist Telkom with the development of a strategy for the migration from classical Intelligent-Network (IN) based services to future broadband services.

2.3.2 Service Implementation on the SATINA Platform

Implementing services on the SATINA platform requires the development of service independent and service specific computational components. The components are defined in the Service Architecture and interact with each other using CORBA to initiate access sessions and launch services. The interfaces and methods that pertain to each object are defined according to the reference point in question. As shown previously, figure 2.4 depicts two key domains: a *user* and *provider* domain. When detailing interactions between the Consumer and the Retailer, the Consumer resides within the *user* domain whilst the Retailer occupies the *provider* domain. Interactions between these two domains are therefore specified according to the Retailer Reference Point (Ret-RP).

- *Initiating an access session*

To initiate an access session, the Consumer requires the asUAP and PA components to be resident on their end-terminal. All Consumer interactions are assumed to take place through a web browser as the user-interface (thus supporting user mobility). To the Consumer, the Retailer represents another Internet web site that they can view. By viewing the Retailer's site, the Consumer has the facility of logging onto an existing account or subscribing as a new user.

On accessing a designated Retailer web-server, the PA (which can contain the asUAP) is downloaded to the Consumer's browser. The PA component takes the form of a digitally signed applet. Applets downloaded from a web server are highly restricted in terms of the operations that they can perform on the client's machine. This 'sandbox' feature is necessary to restrict the applets from performing illegal operations and warding off virus attacks [17]. The PA however needs to act as both a client and a server, communicating with other components via the ORB. The applet housing the PA needs to be digitally signed to allow it to operate outside the bounds of the sandbox.

The downloaded PA prompts the Consumer to enter the required name and password which in turn is passed on to the Initial Agent (IA) for authentication. Once the Consumer is authenticated, their appropriate User Agent (UA) is launched with the Consumer having the ability to list subscribed services, discover new services, view billing and account status, and start a service. All these interactions take place with the Consumer using the browser as a universal GUI. An access session between the Retailer and the Consumer now exists.

- *Providing service functionality*

Service functionality is provided by the service-specific components, namely the ssUAP, USM and SSM. On launching a service, the PA requests the UA to locate the SF that initiates the appropriate USM and SSM.

A ssUAP (containing GUI functionality etc.) also needs to be obtained and initiated. Obtaining the ssUAP can be done dynamically by downloading it when the Consumer subscribes to that specific service. Also, interfaces to the USM need to be relayed back to the ssUAP so that appropriate CORBA methods can be invoked. This process is done via the PA during session setup.

The service can now be executed with the USM and SSM providing the service functionality, and the ssUAP interfacing with the Consumer. Streaming of data, audio, or video takes place between a source and the user's ssUAP under the control of the Communication Session Manager (CSM). The CSM is controlled by the SSM, but this process is out of the scope of this report.

2.3.3 SATINA Network Layout

The SATINA Network (figure 2.7) consists of a number of terminals that are located on three separate Local Area Networks (LAN). These LANs represent three of the main role-players in the TINA Business Model (Consumer, Retailer, and Third Party Provider) and are interconnected via an ATM core network. This structure allows for the logical separation of the main business entities involved, and emulates location transparency that will occur on real networks.

This project utilises the following software and hardware on each LAN (see section 3.2 for a detailed breakdown):

LAN A: Consumer terminal comprising of a web browser and TINA computational components.

LAN B: A Retailer server comprising the TINA Retailer code.

LAN C: An E-commerce application server together with additional TINA Third-Party Provider code.

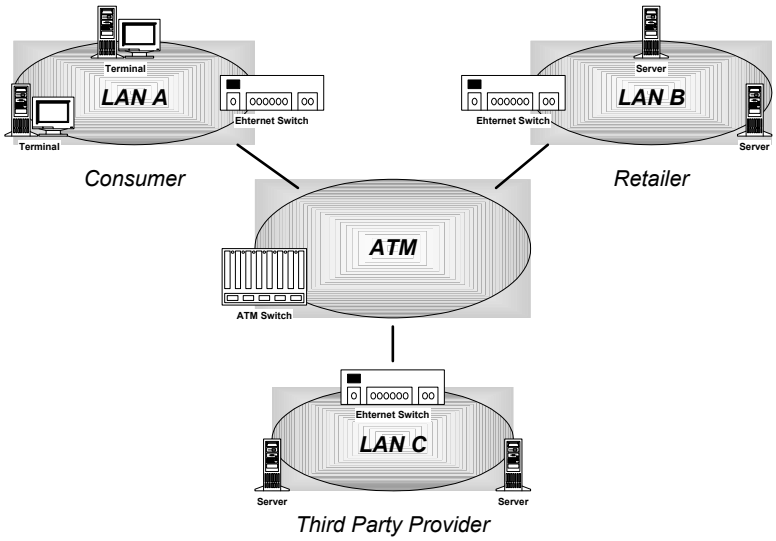

Figure 2.7: SATINA Network Layout

3 TINA E-Commerce Service Design

The design and coding of a prototype proof-of-concept service that allows a TINA Consumer to interface with existing e-commerce application servers involves the implementation of the TINA Business Model within the Internet environment. This process will require the e-commerce provider to act as the 3rd Party Content Provider within the TINA framework, and thus adhere to the 3Pty Reference Point. Some additional functionality would therefore be required by the e-commerce provider to make their service available to these TINA Consumers, whilst still allowing access to normal Internet users.

With a number of prospective e-commerce providers acting as 3rd Party Providers, a suitable TINA service that mirrors current trends in Internet-based e-commerce would be of the form of a Virtual Shopping Mall (VSM). This VSM service would allow the TINA Consumer the ability of performing online transactions as done on the Internet today, but with a number of additional features.

3.1 Design of a Prototype VSM Service

3.1.1 Service Functionality

The VSM service has been designed to allow the access of a TINA Consumer to the e-commerce environment without disrupting access of traditional Internet users. The following functionality has been identified in the provision of this service:

- E-commerce providers require a mechanism to establish an access session with the Retailer to make their site available to the TINA Consumer. To obtain this mechanism, they are required to register with the Retailer to acquire a password, and to be added to the list of e-commerce stores that the Consumer can choose from.

- On starting the VSM service, the Consumer should be able to select a store from this list of on and offline e-commerce providers.
- Selecting an online store (e-commerce provider that is currently involved in an access session with the Retailer) launches a browser on the Consumer's terminal, and directs the Consumer to the correct site for online shopping.
- When the Consumer is finished with their shopping session, all the billing information pertaining to that session should be relayed to the Retailer for account handling. Normal Internet users must still have their accounts handled by the e-commerce provider in the usual manner.
- A mechanism must in place to allow the Consumer to be notified when a requested offline store becomes available for transactions.
- The Consumer must be able to view their billing record from previous purchases, including the time, date, item description, item quantity, and item price.

3.1.2 Application of the TINA Business Model on the Internet

To allow a TINA Consumer access to online shopping, the e-commerce provider requires additional functionality to play the 3rd Party Provider role (and thus adhere to the 3rd Party Reference Point – 3Pty).

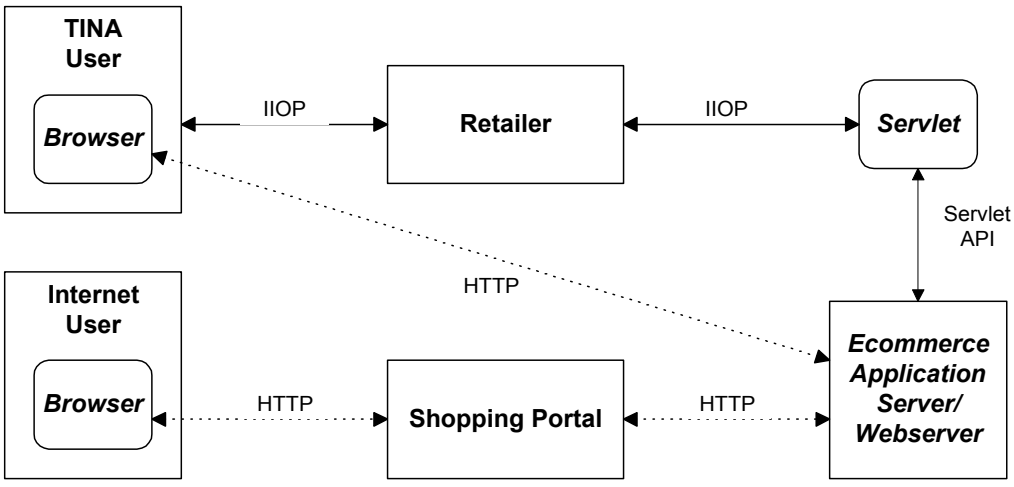

Figure 3.1 Incorporation of the TINA Business Model

TINA-C has not fully specified the 3rd Party Reference Point for interworking between the Retailer and the Third Party Provider. For the VSM service, a number of requirements and interactions between the Retailer and the e-commerce provider need to take place:

- An access session between the two domains is required for a Consumer to be informed of the stores that are available for transactions.
- After a transaction is complete, the billing information generated by the e-commerce provider needs to be relayed to the Retailer for account handling.
- The Retailer requires a mechanism for obtaining third-party server information, which includes server availability and server status.

Work on the 3Pty reference point has been carried out by the Eurescom P715 project [18] whereby parts of the Ret-RP have been reused. With the Eurescom scenario, the Third Party Provider contains IA and UA functionality that allow the Retailer to initiate an access session, and reuse a number of the interfaces defined by the Ret-RP.

This technique is however not well suited for the virtual mall application. Due to the large number of possible online stores, server availability problems, and the Retailer providing a billing-type service, it is desirable to have the e-commerce provider invoke the access session. This requirement is made possible by viewing the Content Provider as a *User* and the Retailer as a *Provider*.

To play the Third Party Provider role, the e-commerce provider obtains a servlet from the Retailer that interfaces with their e-commerce application server (figure 3.1). This servlet contains PA functionality allowing the Content Provider to log onto the Retailer using a 'third party' IA and UA in the same manner that the Consumer gains access to the Retailer. During the initial startup of the e-commerce application server, the server administrator logs onto the Retailer thereby making the store 'available for transactions' to the Consumer using the VSM service.

The e-commerce provider is still responsible for providing the typical e-commerce site. Typical functionality includes their back-end catalogues and providing session management for their shopping basket (using cookies or URL rewriting).

3.1.3 Design of the Computational Components

To provide the functionality stated in section 3.1.1 above, the service-specific objects providing the VSM service must be created and integrated into the service architecture on the SATINA platform. The components are shown in figure 3.2 with specific service interactions detailed in section 3.1.5. The service-specific components include:

- ***ssUAP*** - The ssUAP interfaces with the Consumer to provide the necessary VSM GUIs. The ssUAP has standard interfaces for communicating with the PA, and an interface allowing notification requests emanating from the SSM.
- ***USM*** - With the VSM being a single-user service, the USM is not directly utilised. In our implementation, its purpose is to act as a relay mechanism between the ssUAP and the SSM by mirroring interfaces on either side of the object. For multi-user requirements (e.g. e-commerce with audio consultation), a more comprehensive USM will be required.
- ***SSM*** - The SSM provides the key functionality of the VSM service. It contains the methods to locate online and offline stores, receives and processes billing information, and provides notification functionality for stores that move from the offline to online status.

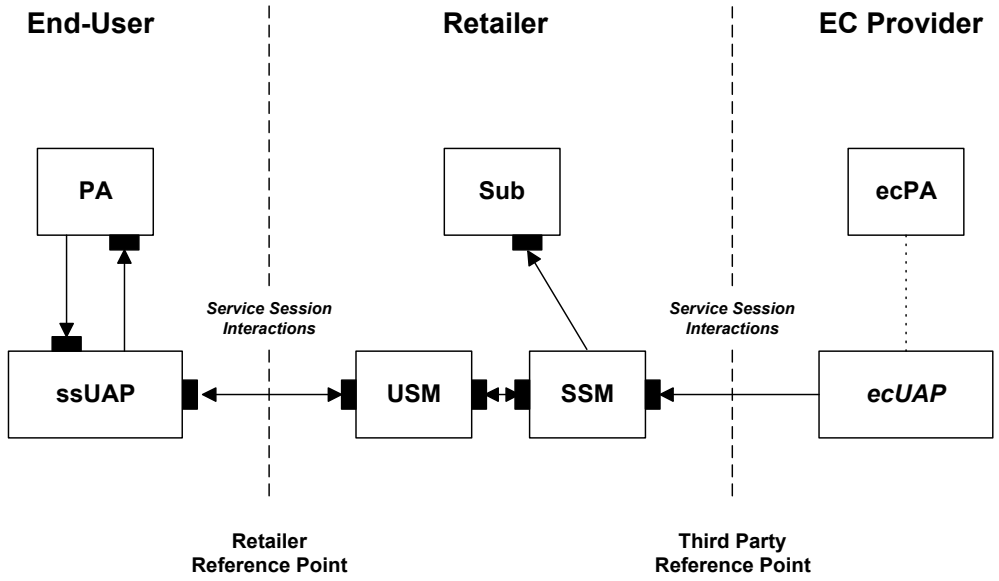

Figure 3.2 Required Computational Components

The remaining component groups that aid in the provision of the service are:

- ***PA*** - Launches the ssUAP and returns the interfaces available on the USM.
- ***SUB*** - Is the subscription component to the Service Architecture. This component provides the list of all users (Consumers and Third-Party Providers) that are permitted to access the Retailer. The list of e-commerce stores can be retrieved from this component.
- ***ecPA*** - Provides the servlet in the Third Party domain with Provider Agent functionality allowing the e-commerce provider to establish an access session with the Retailer.
- ***ecUAP*** - Functionality within the servlet that allows the e-commerce provider to bind to the Consumer's SSM to upload billing information.

Adding the VSM service to the list of offered Retailer services is done using the Retailer administration console which was developed for the SATINA platform.

3.1.4 Service-specific IDL

The service-specific interfaces that are available across the Ret and the 3Pty Reference Points (figure 3.3) have been divided into two modules: Ret: *VSMRet* module, 3Pty: *VSMThirdParty* module. The interfaces within these modules (see Appendix A for corresponding IDL) are either available on the *User* or on the *Provider* side of the reference point, and provide the following functionality (detailed usage scenarios are presented in section 3.1.5):

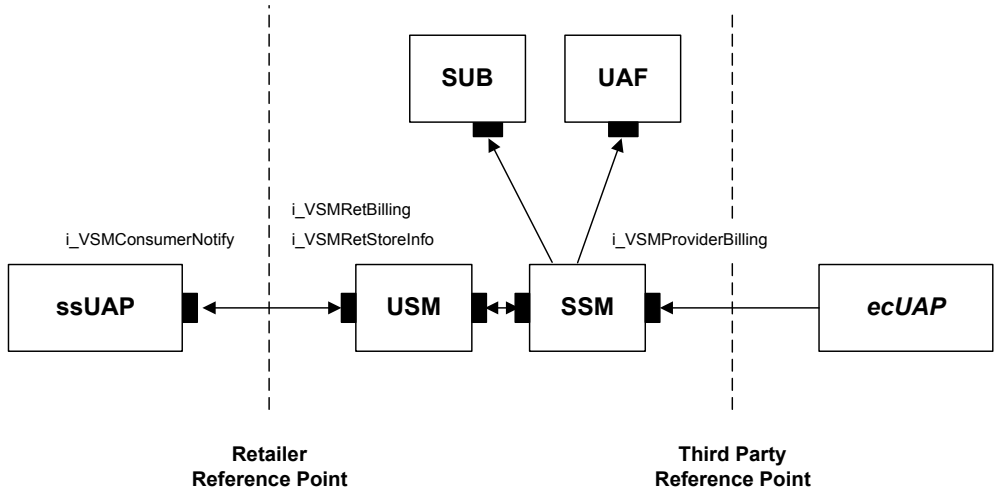

Figure 3.3 Object Interfaces

- Interface i_VSMRetStoreInfo

getAllStores: From **ssUAP to USM to SSM**. SSM calls a method on the SUB to return a list of all e-commerce stores that are registered with the Retailer. (Registration takes place offline by adding the EC store to the database accessed by the SUB).

getOnlineStore: From **ssUAP to USM to SSM**. SSM calls a method on the User Agent Factory (UAF) to determine which e-commerce store is currently involved in an access session (i.e. online).

requestOfflineStore: From **ssUAP to USM to SSM**. SSM spawns a separate thread that periodically executes *getOnlineStore* to poll for a requested e-commerce store that is offline. If the store comes online, the method *notifyOnlineStore* on the interface *i_VSMConsumerNotify* is called.

cancelRequestOfflineStore: From **ssUAP to USM to SSM**. Cancels the thread created in *requestOnlineStore.*

.

- Interface i_VSMRetBilling

viewBilling: From **ssUAP to USM to SSM**. The SSM queries the Consumer's file (using their ID tag) and retrieves the stored billing info.

updateBilling: From **ssUAP to USM to SSM**. Allows the Consumer to pay their account, thus updating the records on the Retailer.

- Interface i_VSMConsumerNotify

notifyOnlineStore: From **SSM to USM to ssUAP**. The thread generated in the method *requestOnlineStore* calls this method to inform the Consumer that a requested offline store is now available.

- Interface i_VSMProviderBilling

addBilling: From **Servlet to SSM**. Allows the e-commerce provider to upload billing data to the Consumer's account.

3.1.5 VSM Service Session Descriptions

- *Starting the VSM Service*

Once the Consumer has established an access session with their Retailer, the Consumer is able to list and start services to which they have subscribed. The VSM service is launched in the following manner:

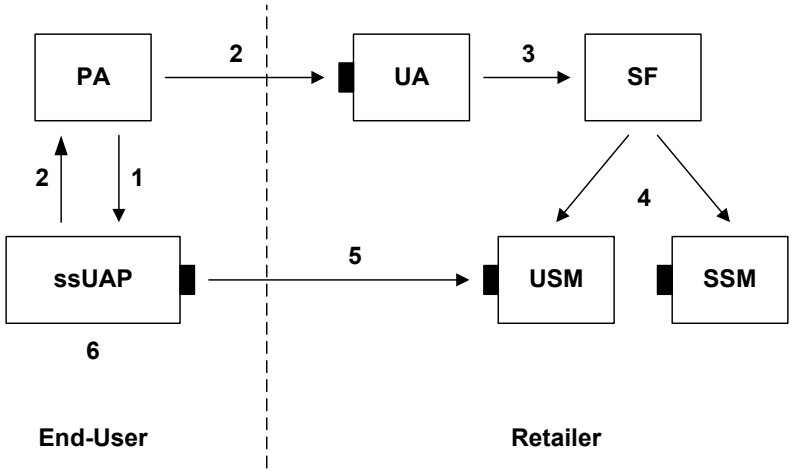

Figure 3.4 Starting the VSM Service

1. The Consumer selects 'Shopping Mall' from the list of subscribed services, resulting in the VSM ssUAP being launched.
2. The method *startService* is executed on the on the PA which in turn is executed on the UA (*interface:* i_RetailerNamedAccess) with the service properties (TINA specific properties) for the VSM.
3. The UA creates a session with the Shopping Mall SF.
4. The SF launches the USM and SSM, which contains the billing and control functionality for the service. USM Interfaces are returned to the ssUAP via the PA.
5. The method *registerInterfaces* is executed on the USM (using the interface that was returned from the USM) to register the ssUAP's interfaces with the USM and SSM.
6. The VSM startup GUI is launched (figure 3.6).

The sequence diagram depicting this scenario is:

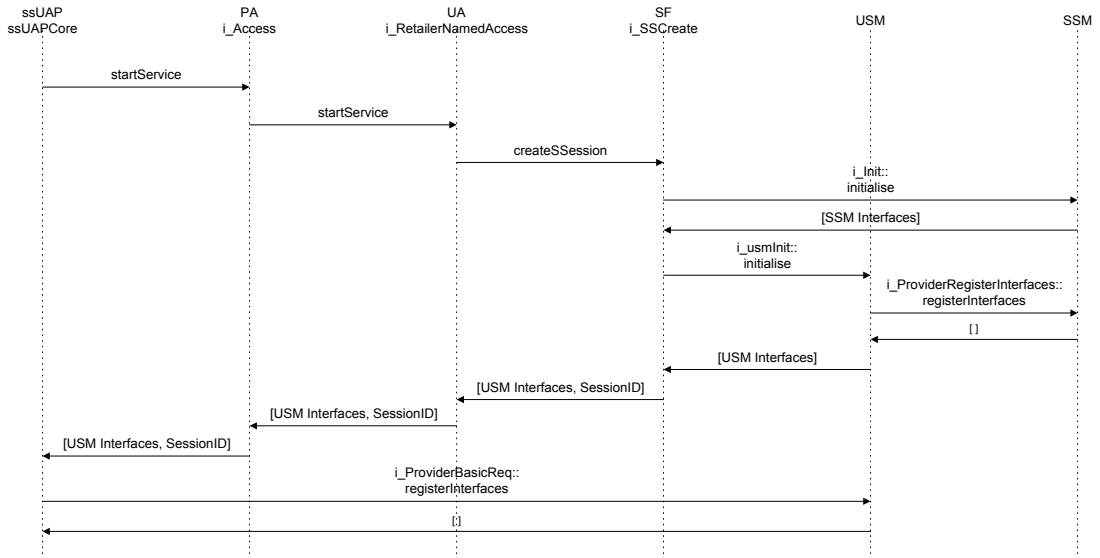

Figure 3.5 Starting the VSM Service (Sequence Diagram)

A screenshot of the startup GUI for the VSM service is shown below. This GUI allows the Consumer to begin shopping, or to view previously stored billing records.

Figure 3.6 Starting the VSM Service (GUI)

- *Initiating a shopping session*

Clicking on the *Shopping* icon initiates a shopping session. This session presents the Consumer with a list of e-commerce stores that have a listing with the Retailer. Stores that are currently available for transactions (those that have established an access session) are identified (highlighted in green in our implementation), with stores that are currently unavailable, highlighted in red.

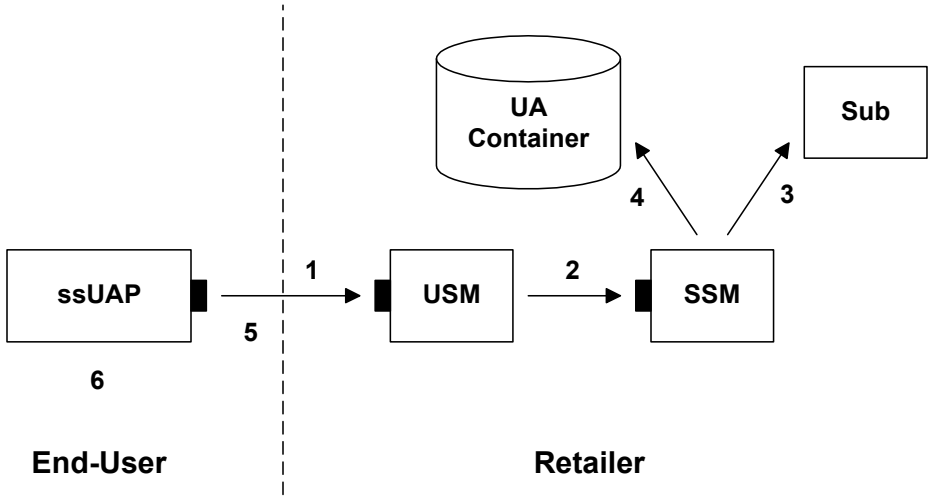

Figure 3.7 Initiating a Shopping Session

1. The methods *getOnlineStores* and *getAllStores* are executed on the USM (*interface:* i_VSMRetStoreInfo).
2. The USM inturn relays the same requests to the SSM.
3. The list of all e-commerce stores that have a listing with the Retailer is retrieved from the SUB component (*interface:* i_Subscribe and i_SubscriberInfoMgmt).
4. Each store retrieved from the SUB component is compared to the list of e-commerce providers that are currently involved in an access session with the Retailer (this list is maintained in a UA Container object that contains all currently active UAs).
5. The list of online and offline stores are returned to the ssUAP via the USM.
6. A GUI is presented to the Consumer detailing the available stores (figure 3.9).

The sequence diagram depicting this scenario is:

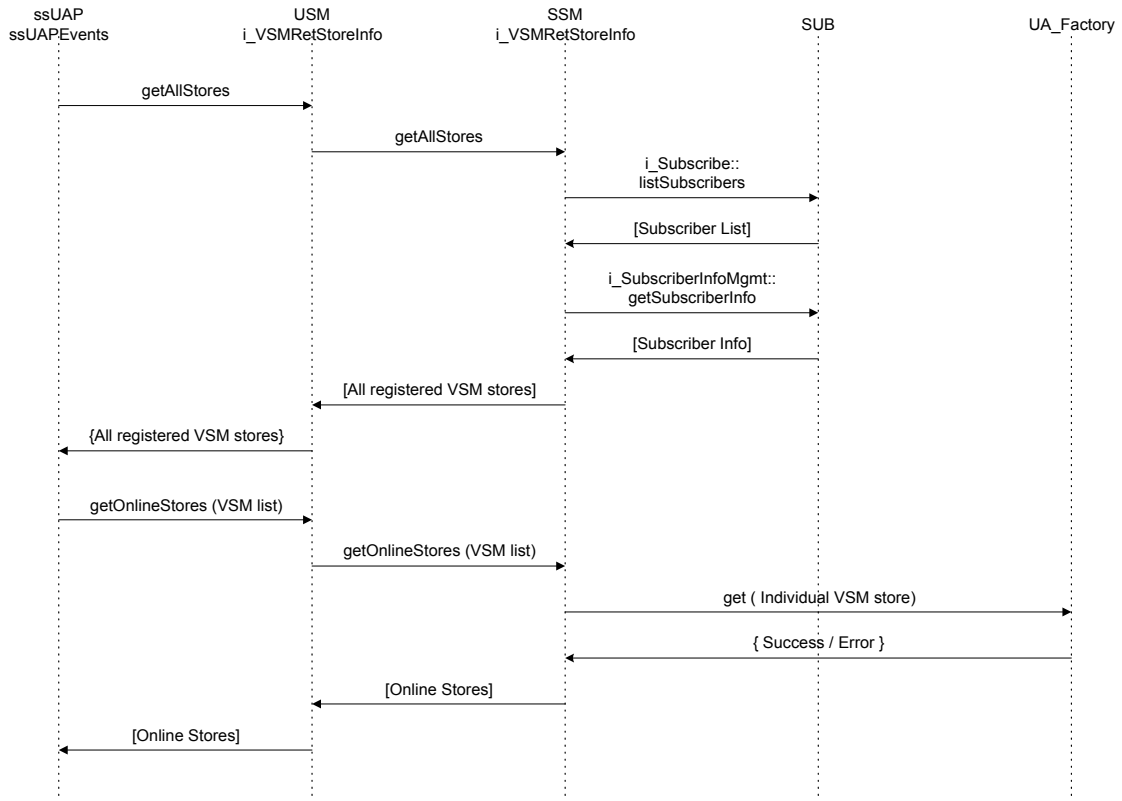

Figure 3.8 Initiating a Shopping Session (Sequence Diagram)

The screenshot depicting the list of available online and offline stores is shown below:

Figure 3.9 Initiating a Shopping Session (GUI)

- *Purchasing scenario*

Clicking on an available store (depicted green to the Consumer in our implementation) prompts the launch of a browser that is directed to the corresponding e-commerce site (the URL of the site is included in the store information previously returned to the ssUAP). The Consumer then browsers the e-commerce site in the usual HTTP manner, selecting desired goods and placing them in the typical shopping basket. When the final 'purchase button' is activated, the e-commerce application server interrogates the HTTP request for the presence of an IOR (sent as a variable in the URL) to determine whether the end-user is a TINA or Internet user. If an IOR is present, the application server uses it to upload the billing information back to the Consumer's account held with the Retailer. Normal Internet users are still directed to the site's own account handling system.

Security aspects relating to this approach are presented in section 3.2.3.

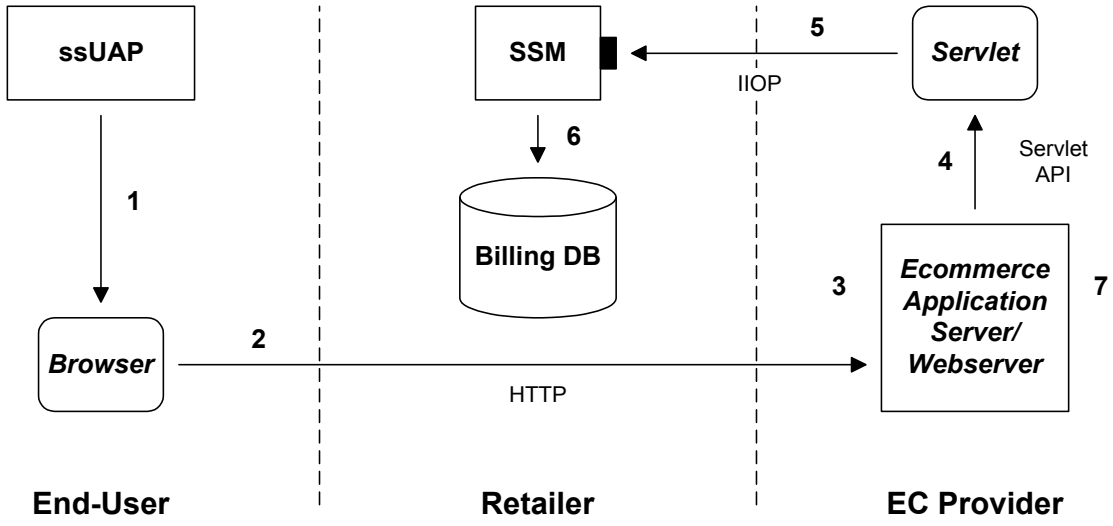

Figure 3.10 Purchasing Scenario

1. On selecting an online store from the presented list, an instance of a browser is launched.
2. The browser is directed to the corresponding URL specified by the ssUAP, with the IOR of the Consumer's SSM (*interface:* i_VSMProviderBilling) added to the HTTP request.
3. The desired goods are chosen in the usual Internet (HTTP) e-commerce manner.
4. When the final purchase button is activated, the application server detects that the user is a TINA Consumer (due to the presence of the IOR) and passes the billing information together with the SSM's IOR to the servlet.
5. The servlet attempts to bind to the SSM using the IOR.
6. The method *addBilling* is executed to upload the billing information to the database in the Retailer.
7. Successful addition to the database is reflected back to the Consumer via the webserver.

The sequence diagram depicting this scenario is:

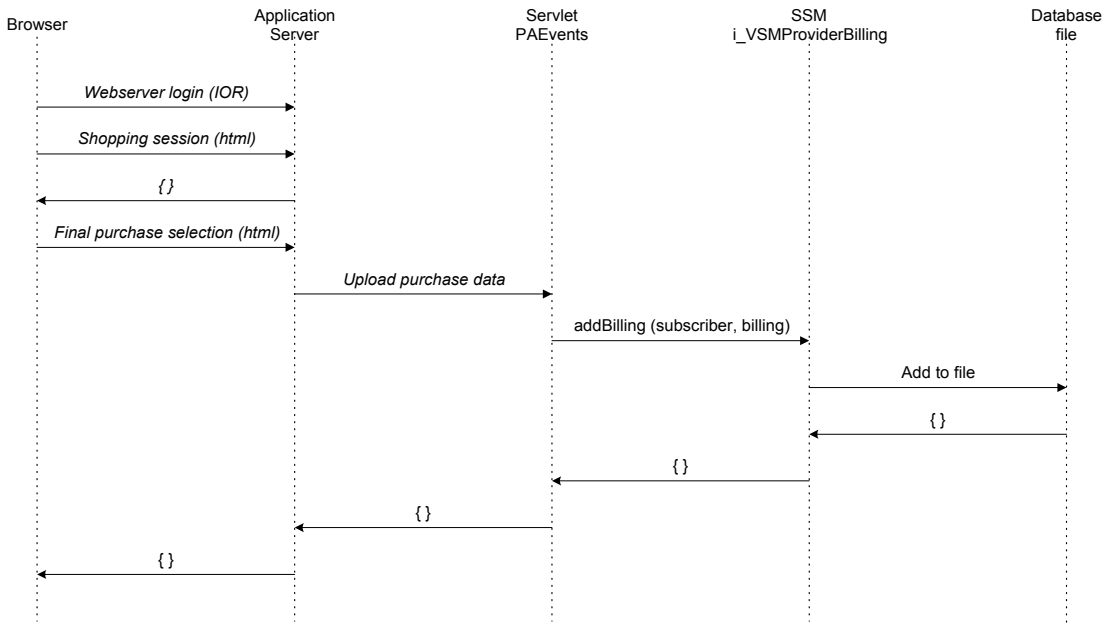

Figure 3.11 Purchasing Scenario (Sequence Diagram)

A screenshot of the Enhydra Online Golf Store (refer to section 3.2), which is used to demonstrate the e-commerce purchasing scenario, is shown below:

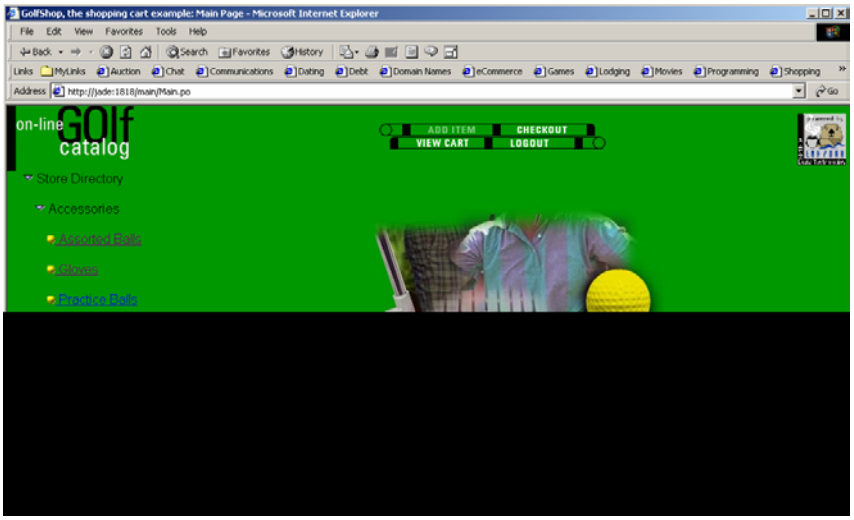

Figure 3.12 Purchasing Scenario (GUI)

- *Call-back scenario*

On selecting an e-commerce store that is offline (depicted red to the Consumer in our implementation), the Consumer can request to be notified when the selected store becomes available. This request can be cancelled at any time by the Consumer, or when the Consumer ends their access session with the Retailer.

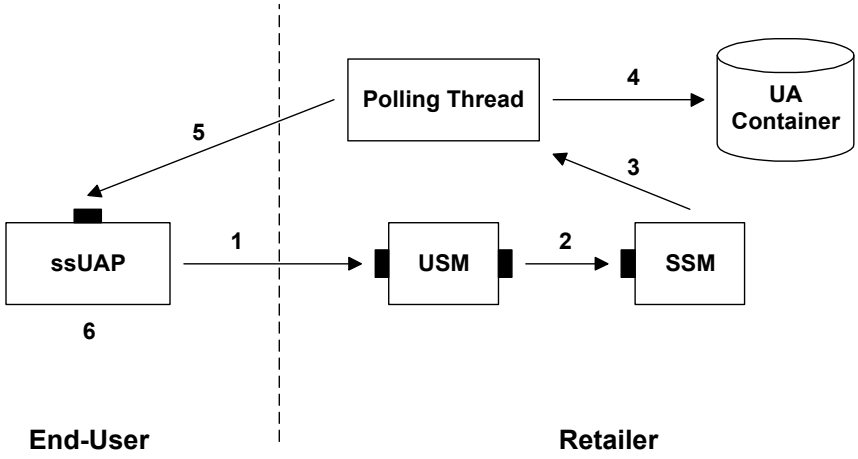

Figure 3.13 Call-back Scenario

1. On selecting an offline store from the presented list, the method *requestOfflineStore* is called on the USM (*interface: i*_VSMRetStoreInfo).
2. The USM inturn relays the same request to the SSM.
3. The SSM creates a new polling thread within the Retailer, and returns a success notification to the ssUAP.
4. The polling thread periodically interrogates the UA Container to determine if the requested store has established an access session with the Retailer (has returned online).
5. When the requested store becomes available, the thread binds to the Consumer's ssUAP (*interface:* i_VSMConsumerNotify) and calls the method *notifyOnlineStore.*
6. A GUI is launched to notify the Consumer of the available store.

The sequence diagram depicting this scenario is:

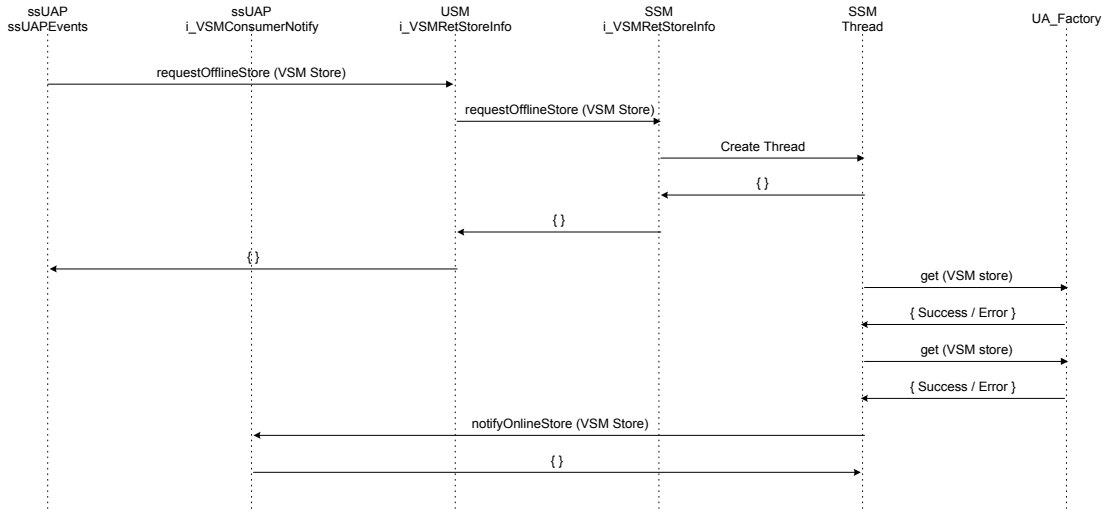

Figure 3.14 Call-back Scenario (Sequence Diagram)

Figure 3.15 Call-back Scenario (GUI)

- *Retrieving billing records*

The billing records from previous e-commerce shopping sessions can be viewed by selecting *View Billing* from the VSM startup menu (figure 3.6).

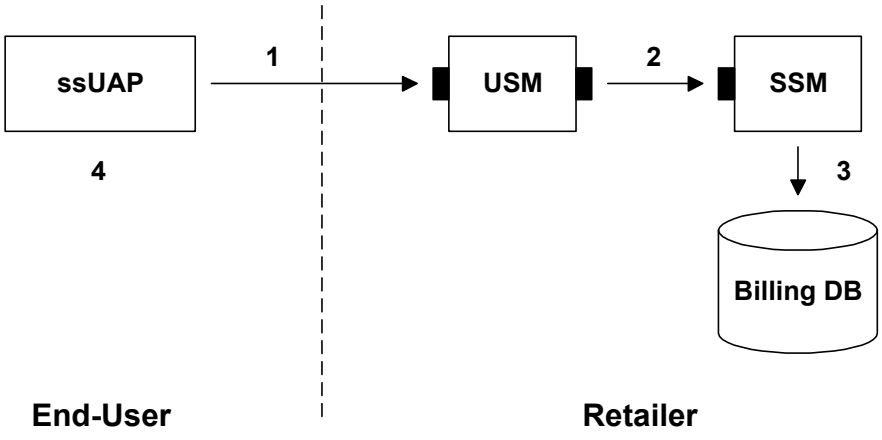

Figure 3.16 Viewing Billing Records

1. The method *viewBilling* is called on the USM (*interface:* i_VSMRetBilling).
2. The USM inturn relays the same request to the SSM.
3. The SSM retrieves the billing information relating to the Consumer.
4. A billing GUI is launched (figure 3.18) allowing the Consumer to view and pay the separate accounts. (Paying the account mealy deletes the entry in this proof-of-concept implementation)

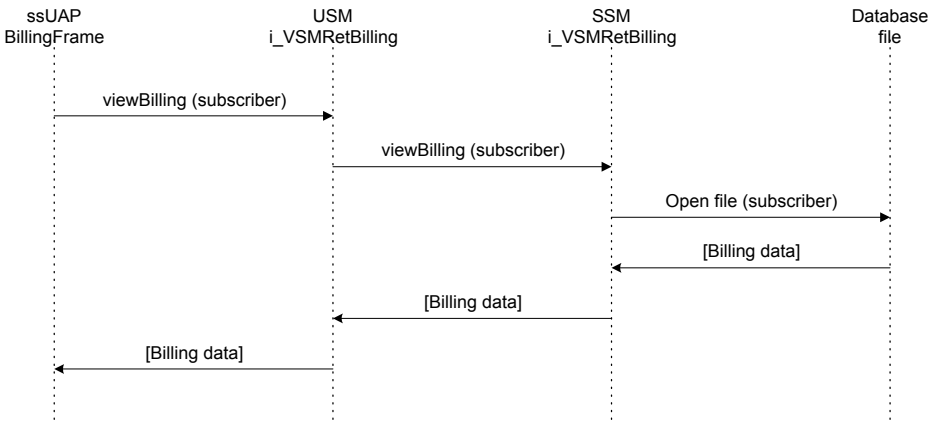

Figure 3.17 Viewing Billing Records (Sequence Diagram)

A screenshot depicting the billing GUI is shown below:

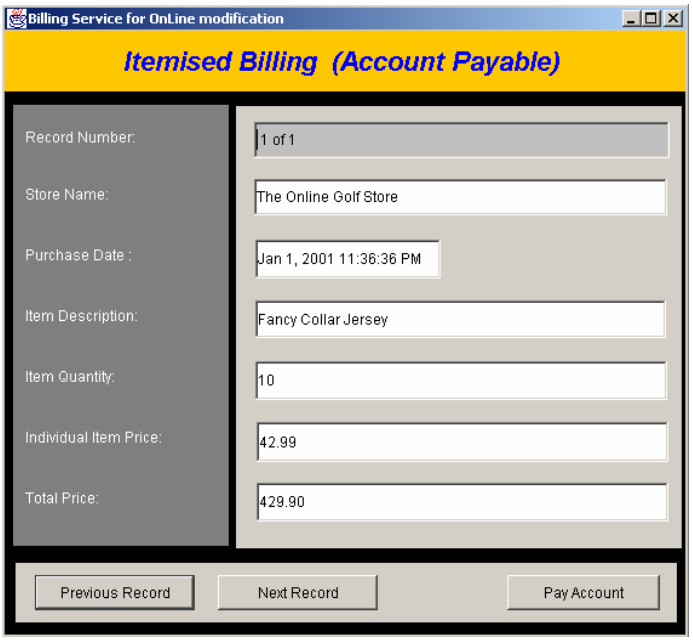

Figure 3.18 Viewing Billing Records (GUI)

3.2 Prototype Implementation

3.2.1 Computational Components

The Service Architecture in the SATINA trial has been realised by the implementation of the various computational components. The Consumer domain objects (PA, and the various ssUAP's, including the VSM ssUAP) have been implemented using the Java programming language (jdk 1.2.x) [19] and executes on both a Linux and Windows NT operating system. The ORB on the Consumer domain is implemented using Orbacus 3.1.3 for Java [20]. The GUI's for the VSM service are designed using JBuilder 3.0 [21].

The Retailer domain components (IA, UA, SUB, SF, SSM and USM) have been implemented in C++ and execute on a Linux operating system. MICO 2.2.7 [22] is chosen as the ORB for the C++ implementation.

3.2.2 E-Commerce Provider Implementation

The application server Lutris Enhydra 3.0 [23] was obtained to act as the typical e-commerce server. Enhydra is an application server for running robust and scalable web applications, and provides dynamically generated content for the system's webserver. Enhydra supports the Java Servlet 2.2 API (a necessary requirement for our solution), and provides all the typical e-commerce capabilities including session management to deliver content to the end-user's browser.

An Enhydra e-commerce service in the form of an online Golf Store is used as the demonstration e-commerce store.

3.2.3 Evaluating the Prototype VSM Service

The prototype service makes use of servlet technology to provide TINA Third Party functionality to the e-commerce provider. Servlet technology was chosen due to the wide-scale support by most e-commerce application servers. This technology allows CORBA functionality to be included on the provider side and hence allow the e-commerce server to interface with the TINA Retailer. This approach proved suitable for meeting the stated problem requirements.

Security is an important issue when considering online e-commerce. The proposed solution provides an alternative approach to security issues since no financial information is exchanged between the Consumer and the e-commerce provider. To fraudulently purchase items, a user would have to establish an access session with the Retailer and launch the VSM service. This process creates the SSM and passes the IOR onto the e-commerce provider. The SSM (and the issued IOR) is only *active* whilst the VSM service is *active*, and billing information can *only* be uploaded to the Retailer during this period. Thus no purchasing can be done unless a TINA Consumer has launched the VSM service (implying that the TINA Consumer is involved in a secure access session with the Retailer). With the implementation of the prototype, the IOR is passed to the e-commerce provider with the initial HTTP request. This process serves as a proof-of-concept purpose, but a more secure approach for transferring the IOR might be necessary for commercial implementations.

The shopping mall model also presents a number of solutions to the problems that were presented in section 2.2.1 The VSM model allows for:

- Centralised billing - All billing is handled by the Retailer thus providing a convenient centralised billing mechanism.
- Increased security as a result of the Consumer financially interfacing with only one party (the Retailer).
- A call-back mechanism for minimising transaction loss due to server failure
- The possibility of providing global user-profile customisation and management.

4 Conclusion and Future Work

4.1 Review

The project examined the use of TINA concepts to provide the required service architecture layer for the Next Generation Network. In particular, it detailed how the current Internet-based e-commerce environment could be integrated as a service within the TINA NGN solution. A prototype proof-of-concept service in the form of a Virtual Shopping Mall was designed and implemented on the SATINA platform. The service used servlet technology to integrate with a simulated e-commerce environment, and provided the TINA Consumer with shopping capabilities without disrupting access of traditional Internet users.

The model also presented additional features in the form of centralised billing and notification services for the TINA Consumer.

4.2 Conclusions

The process to investigate how web-based e-commerce could be offered as a service on a network that employs the TINA Service Architecture involved identifying the technologies to support this process, analysing the business model implications from providing this service, and demonstrating these concepts by means of a prototype service implemented on the SATINA platform.

From a technology perspective, the VSM service demonstrates how Internet technologies and object middleware can be integrated to provide a number of business benefits. The choice of servlet technology to provide TINA and CORBA functionality on the e-commerce provider's domain proved a viable solution due to its seamless interworking with the prototype e-commerce application server.

The VSM service also adequately demonstrated the business benefits obtained by implementing the TINA Business Model. Both the Consumer and the e-commerce provider were shown to benefit from centralised billing, inherent increased security, and mechanisms for dealing with e-commerce transaction failures.

4.3 Recommendations for Further Work

The design presented in this report can be extended to provide additional functionality and benefits.

Firstly, a mechanism for the dynamic deployment of ssUAPs to the Consumers's terminal has not been implemented within the SATINA platform. This process would ideally be done during the user subscription phase for that particular service. Provisioning of the VSM ssUAP was thus done offline. Another improvement that can be made, is the manner in which the browser resident on the user's terminal is launched. On selecting an available store, the software was hard-coded to launch Internet Explorer with the URL received from the SSM. This code can be improved using the terminal's OS built-in types to launch the system's standard browser.

The servlet interfacing to the e-commerce application server can be extended to allow for server status query (determining total capacity, current load etc.). This feature would allow the Retailer to determine the load on the EC server before directing the consumer for a shopping session. The Retailer would also be able to perform load balancing (at the application layer) if the EC provider has more than one operational server.

Finally, with the EC provider becoming a node on the DPE, the Consumer can request to be pushed information when it becomes available, using CORBA as the signalling mechanism. The Consumer might want to be notified when a requested share price has been reached, or when their bid for an item at an auction site has been exceeded.

References

[1] Ubiquity Software Corporation, “Application Powered Networks with SIP – White Paper”, ‘http://www.ubiquity.net’, 2000.

[2] M. Schenk, G. Jarboe, “OMG White Paper: Open Service Marketplace”, OMG Doc. No. telecom/00-05-02, Version 1.0, 17 May 2000.

[3] TINA-C, “Overall Concepts and Principles of TINA”, Version 1.0, ‘http://www.tinac.com, Feb 1995.

[4] South African TINA Trial, ‘http://satina.wits.ac.za’.

[5] H. Hanrahan, “Convergence, Digitisation and New Technologies: Toward the Next Generation Network”, ICT Conference, 2000.

[6] M. Finkelstein, J. Garrahan, D. Shrader, and G. Weber, “The Future of the Intelligent Network”, Telcordia Technologies, IEEE Communications Magazine, June 2000.

[7] Unisphere Networks, “IP Telephony Protocols Positioning Paper”, ‘http://www.unispherenetworks.com’, 2000.

[8] A. Aslam, “Switching and Signalling – The New Architecture”, London Communications Symposium, ‘http://www.ee.ucl.ac.uk/lcs/papers2000/content.html’, 2000.

[9] A.R. Modarressi, S. Mohan, “Control and Management in Next-Generation Networks: Challenges and Opportunities”, IEEE Communications Magazine, October 2000.

[10] D. Shemony, “Create the basis for new value”, ‘http://www.broadsoft.com’, 2001.

[11] C. Abarca, P. Farley, J. Forslow, J.C. Garcia, T.Hamada, P.F. Hansen, S.Hogg, H. Kamata, L. Kristiansen, C.A. Licciardi, H. Mulder, E. Utsunomiya, M.Yates, “Service Architecture”, TINA-C Baseline, 16 June 1997.

[12] C. Abarca, P. Farley, J.C. Garcia, T.Hamada, P. Hellemans, K. Nakashiro, C.A. Licciardi, M.Yates, “Service Component Specification”, TINA-C Deliverable, 19 January 1998.

[13] Object Management Group, ‘http://www.omg.org’.

[14] R. Rodman, “E-Commerce”, i-Group, ‘http://internetigroup.com/smnarticls5.html’, 1999.

[15] A.T. van Halterenm L.J.M. Nieuwenhuis, M.R. Schenk, M. Wegdam, “Value Added Web: Integrating WWW with a TINA Service Management platform”, TINA Conference, April 1999.

[16] TINAC, “Retailer Reference Point”, Version 1.1, April 1999.

[17] JAVA Sun Security, ‘http://java.sun.com/security’, 2000.

[18] S. Rana, M.A. Fisher, C. Egelhaaf, “Implementation and Interoperability experience with TINA Service Management Specifications”, TINA 1999.

[19] Sun Microsystems, JAVA, ‘http://java.sun.com/j2se’, 2000.

[20] Object Oriented Concepts, Orbacus, ‘http://www.ooc.com’, 2000.

[21] Borland, ‘http://www.borland.com/jbuilder’, 2000.

[22] Mico is CORBA, ‘http://www.mico.org’, 2000.

[23] Lutris, ‘http://www.enhydra.org’, 2000.

Appendix

Appendix A: Definition of the VSM IDL

```
// ******************************************
// File: VSMCommonTypes.idl
// Author: Shaun Brassell
// Date: 13/06/2000
// ******************************************


#ifndef vsmcommontypes_idl
#define vsmcommontypes_idl

#include "TINACommonTypes.idl"

module VSMCommonTypes {

    typedef TINACommonTypes::t_UserId userId;
    typedef string t_estoreName;
    typedef string t_estoreId;
    typedef string t_estoreUrl;
    typedef string t_estoreItemDescr;
    typedef float t_estoreItemPrice;
    typedef long t_estoreItemQty;
    typedef string t_purchaseDate;
    typedef string t_status;

    struct t_estoreData {
         t_estoreId estoreId;
         t_estoreName estoreName;
         t_estoreUrl estoreUrl;
         t_status estoreStatus;
    };

    typedef sequence<t_estoreData> t_estoreList;

    struct t_estoreBilling {
         t_purchaseDate purchaseDate;
         t_estoreName estoreName;
         t_estoreItemDescr estoreItemDesc;
         t_estoreItemQty estoreItemQty;
         t_estoreItemPrice estoreItemPrice;
    };

    typedef sequence<t_estoreBilling> t_estoreBillingList;

};

#endif
```

```
// ******************************************
// File: VSMRet.idl
// Author: Shaun Brassell
// Date: 12/05/2000
// ******************************************

#ifndef vsmret_idl
#define vsmret_idl

#include "VSMCommonTypes.idl"

module VSMRet {

    interface i_VSMRetStoreInfo {

         void getOnlineStores (
              in VSMCommonTypes::userId userName,
              inout VSMCommonTypes::t_estoreList estoreList
         );

         void getAllStores (
              in VSMCommonTypes::userId userName,
              out VSMCommonTypes::t_estoreList estoreList
         );

         void requestOfflineStore (
              in VSMCommonTypes::userId userName,
              in VSMCommonTypes::t_estoreName estoreName,
              out VSMCommonTypes::t_status requestStatus
         );

         void cancelRequestOfflineStore (
              in VSMCommonTypes::userId userName,
              in VSMCommonTypes::t_estoreName estoreName,
              out VSMCommonTypes::t_status requestStatus
         );

    };

    interface i_VSMRetBilling {

         void viewBilling (
              in VSMCommonTypes::userId userName,
              out VSMCommonTypes::t_estoreBillingList estoreBillingList,
              out VSMCommonTypes::t_status viewStatus
         );

         void updateBilling (
              in VSMCommonTypes::userId userName,
              in VSMCommonTypes::t_estoreBillingList estoreBillingList,
              out VSMCommonTypes::t_status updateStatus
         );

    };



    interface i_VSMConsumerNotify {
```

```
        void notifyOnlineStore (
             in VSMCommonTypes::t_estoreName estoreName,
             out VSMCommonTypes::t_status notifyStatus
        );

    };

};

#endif
```

```
// ******************************************
// File: VSMThirdParty.idl
// Author: Shaun Brassell
// Date: 13/06/2000
// ******************************************

#ifndef vsmthirdparty_idl
#define vsmthirdparty_idl

#include "VSMCommonTypes.idl"

module VSMThirdParty {

    interface i_VSMProviderBilling {

         void addBilling (
              in VSMCommonTypes::userId userName,
              in VSMCommonTypes::t_estoreBillingList estoreBillingList,
              out VSMCommonTypes::t_status addStatus
         );


    };


};

#endif
```

Appendix B: Description of TINA Computational Components

Each component in the Service Architecture carries out a certain responsibility for supporting access, service, or communication sessions. The components are defined as follows:

THE ACCESS SESSION RELATED COMPUTATIONAL COMPONENTS

as-UAP – Access Session User Application allows a human user, or another application to make use of the capabilities of the PA through an interface to the user. It acts as the end point supporting only access related capabilities

PA – Provider Agent supports a user accessing their UA and making use of services. It is a service independent component defining the user's end point of an access session.

IA – Initial Agent is a user and service independent component that supports capabilities to authenticate the requesting domain and establishing access sessions.

UA – User Agent is a service independent component representing the user in a service provider domain. The UA controls and manages the user and service session.

SUB – Subscription Management is a service independent component that mangers the subscribers, subscriptions and users for a whole set of services.

THE SERVICE SESSION RELATED COMPUTATIONAL COMPONENTS

ss-UAP – Service Session User Application interacts with the PA to start, resume or join a service session. It interacts with the USM to perform all service specific session control, and deals with the adding or modifying of stream bindings.

SF – Service Factory is a service specific object that creates and manages the service session. It also assembles the resources necessary for the created component.

SSM – Service Session Manager is created by a SF to support the adding, inviting and removal of users to a service session. It also manages the resources shared among users.

USM – User Service Session Manager is created by the SF to support interactions between the access and service sessions through interactions with the UA.

CSM – Communication Session Manager provides connectivity functions to the SSM. It manages requests for connections (streams) from applications and uses a *connection coordinator (CC)* to handle the network complexity, bandwidth and QoS.

TCSM – Terminal CSM is part of the user's terminal. It manages the communication characteristics of the terminal and controls the bindings within the user's domain.

Appendix C: CORBA Overview

The Common Object Request Broker Architecture (CORBA) is a standard for designing distributed applications, and forms a key part of the Object Management Architecture (OMA). CORBA was developed by the Object Management Group (OMG) and is an object-oriented approach to the problem of distributed systems. CORBA provides a peer-to-peer model and enables full transparency over a wide range of programming languages and operating systems.

- Interface Definition Language

Location and programming language transparency is obtained by the use of OMG's Interface Definition Language (IDL). IDL is used to define interfaces, their attributes, methods, and parameters to those methods within the interface. IDL is the means by which a particular object implementation tells its potential clients what operations are available and how they should be invoked. From the IDL definitions, it is possible to map CORBA objects into particular programming languages or object systems.

- The Object Request Broker

The central component of CORBA is the Object Request Broker (ORB). This component provides the communication infrastructure needed to identify and locate objects, handle connection management, deliver data, and request communication. One CORBA object never talks directly with another. Instead the object requests for an interface to the ORB running on the local machine. The local ORB then passes the request to an ORB on the requested machine. The remote ORB locates the appropriate object and passes back an object reference to the requester.

The functions of the ORB are as follows:

- Lookup and instantiate objects on remote machines.
- Marshal parameters from one application object to the other.
- Handle security issues across machine boundaries.
- Retrieve and publish data on objects on the local machine for another ORB to use.
- Invoke methods on a remote object using static method invocation.
- Invoke methods on a remote object using dynamic method invocation.
- Automatically instantiate objects that are not currently running.
- Route callback methods to the appropriate local object being managed.
- Communicate with other ORBs using the Internet Inter-ORB Protocol (IIOP).

The figure below shows a request being sent by a client to an object which is implemented on the server. The client is the entity that wishes to perform an operation on the object, and the object implementation is the code and data that actually implements the object.

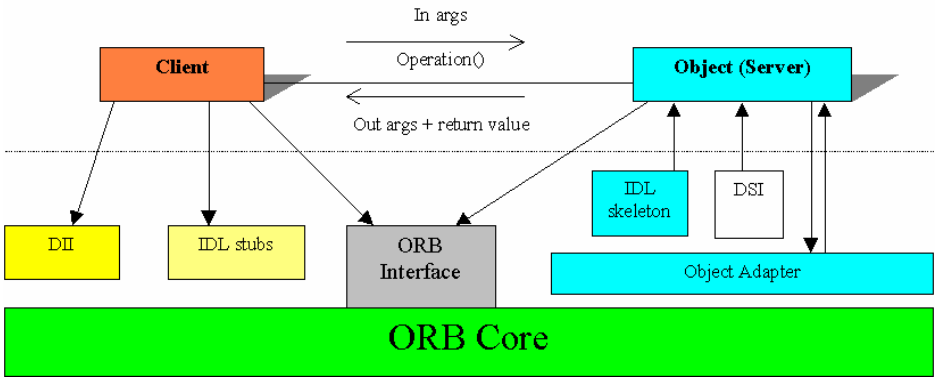

Figure: Object Request Broker Structure

To make a request, the client can use the Dynamic Invocation Interface – DII (an interface that is independent of the target object's interface) or an OMG IDL stub (specific to the target object's interface). The client can also directly interact with the ORB for miscellaneous services.

The ORB locates the appropriate implementation code, transmits parameters, and transfers control to the object implementation through an IDL skeleton (specific to the object implementation) or though the Dynamic Skeleton Interface – DSI (a generic interface). In performing the request, the object implementation may obtain some services from the ORB through the Object Adapter.

Services provided by the ORB through an Object Adapter often include: generation and interpretation of object references, method invocation, security of interactions, object and implementation activation and deactivation, mapping object references to implementations, and registration of implementations. The wide range of object granularities, lifetimes, policies, implementation styles, and other properties make it difficult for the ORB Core to provide a single interface that is convenient and efficient for all objects. Thus, through Object Adapters, it is possible for the ORB to target particular groups of object implementations that have similar requirements with interfaces tailored to them. When the request is complete, control and output values are returned to the client.

Appendix D: Project CD Description

A CD has been included within this document that contains TINA documentation, generated code, and applications that were used during this project. The figure below details the directory structure of the CD:

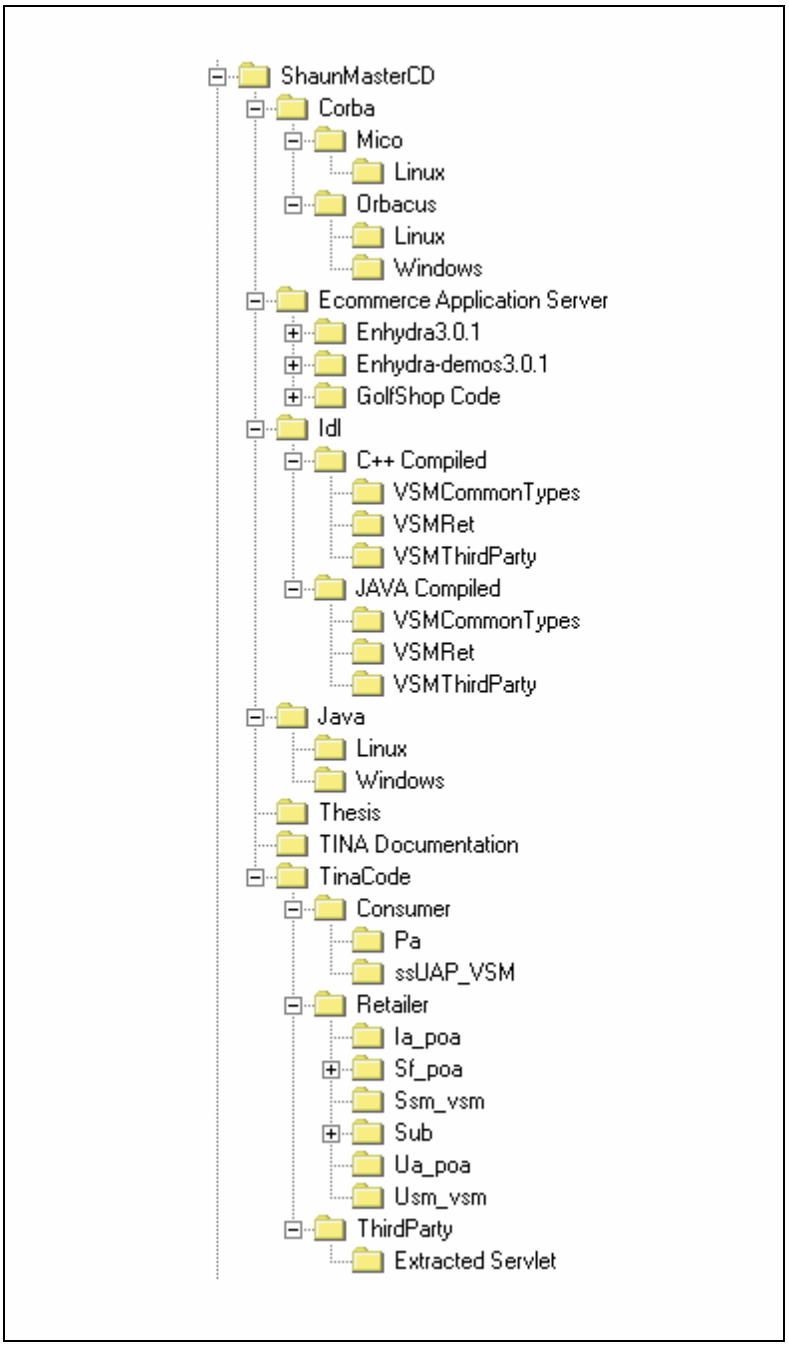

Figure: CD Directory Structure

The seven main directories contain the following programs:

- ***Corba***

  Compressed installation code for Mico and Orbacus (Linux and Windows NT).

- ***E-Commerce Application Server***

  Enhydra Application Server together with the GolfShop demonstration code.

- ***IDL***

  The IDL for the VSM service (with JAVA and C++ compilations).

- ***JAVA***

  Installation files for JAVA 1.2.x (Linux and Windows NT).

- ***Thesis***

  Electronic copy of this document.

- ***TINA Documentation***

  Relevant TINA documentation from the TINA-C (PDF format).

- ***TINACode***

  Directory contains the TINA code obtained and created for the VSM service. The directory is divided into the Consumer, Retailer, and Third Party Provider domains, and contains the implementation of the relevant computational components.