

**Department of
Computer
Science**

**MSc. I.T. Project Report
2002**

**Mobile Software
Development for
an Open Source
E-Learning
Platform**



Author
Erlina Cut-Hennies

Supervisor
Dr. Alan Pearmain

Report: 30 August 2002

Disclaimer

Name: Erlina Cut-Hennies

Title: “Mobile Software Development for an Open Source E-Learning Platform”

This report is submitted as part requirement for the degree of MSc in Information Technology at the University of London. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Acknowledgement

I would like to express my gratitude to my supervisor Dr. Alan Pearmain for his advice and guidance during the project. Also, I would like to thank Lofi Dewanto for testing this program on the OpenUSS server and his advice on the program. Thank you to my husband and my family for their love and support. And last but not least, I would like to thank all the people who supported me.

Abstract

This project adds a Wireless Application Protocol (WAP) presentation interface to the Open University Support System (OpenUSS), which can be categorised as a Learning Management System. This supports mobile access to teaching information in a university. The objective of this project is to design a mobile access that provides OpenUSS users with information retrieval via mobile devices from the universities that use OpenUSS system.

OpenUSS was implemented purely in Java using Java2 Enterprise Edition (J2EE). Business processes within the learning and teaching domain are implemented with server-side components Enterprise JavaBeans (EJB). As an EJB-container, the Open Source product JOnAS is used. To provide the mobile access in accordance with Java technology, Wireless Markup Language (WML) and Java to produce servlets are applied for the implementation. A servlets container named Enhydra, which is also Open Source is utilised to enable a dynamic WAP-page.

Presentation technologies other than WAP such as i-mode, Java2MicroEdition (J2ME), as well as the existing bearer services of WAP such as Circuit-Switched Data Transmission Services, 2.5 G (General Package Radio Service), and 3 G (Universal Mobile Telephone System) are examined briefly.

Testing the implementation is done using a development kit that includes a WAP simulator and WAP gateway simulation. Using local installation, the WAP-Browser is able to retrieve the same information as on the OpenUSS-Webapplication. All the problems during the course of testing and their solution are illustrated deeply. An evaluation of the system demonstrated that clients of university's systems of OpenUSS use it successfully. Once the system is open and integrated on the web, many universities that already use the system of OpenUSS will benefit from this project.

TABLE OF CONTENTS

0 Introduction 6

1 Open Source Software and Open University Support System 6

1.1 Open Source Software 6

1.2 Open University Support System..... 8

 1.2.1 OpenUSS as an Open Source Learning Management System..... 8

 1.2.2 Technical Background of OpenUSS..... 11

2 Wireless Application Protocol and Wireless Markup Language 12

2.1 Introduction to Wireless Application Protocol 12

2.2 Security in the Wireless Application Protocol 15

2.3 The Wireless Markup Language..... 15

2.4 WAP as an Independent Protocol and the Bearer Services..... 16

 2.4.1 Short Message Service..... 17

 2.4.2 Circuit-Switched Data Transmission Services 17

 2.4.3 General Packet Radio Service 17

 2.4.4 Universal Mobile Telephone System..... 18

2.5 Presentation Techniques: i-mode versus WAP 19

3 Java 2 Micro Edition, Java 2 Enterprise Edition and the Products 20

3.1 Java2MicroEdition (J2ME) 20

3.2 Java™ 2 Platform, Enterprise Edition Specification 22

3.3 Enterprise JavaBeans 23

3.4 The Products of J2EE and the OpenUSS Architecture..... 24

4 OpenUSS-WAP Software Development..... 31

4.1 Requirement Analysis and Use Cases..... 31

4.2 Activity Diagram 32

4.3 Class Diagrams 34

4.4 Design and Implementation 37

4.5 Testing and Evaluation 45

5 Future Development 47

6 Conclusion..... 47

Bibliography 48

0 Introduction

Nowadays, high quality teaching needs a modern infrastructure to establish an efficient teaching and learning process. To improve the presence of university foundations based on the Internet, a software package named OpenUSS (Open University Support System) was developed as an Open Source platform to support e learning for universities.

OpenUSS is an existing open source e-learning platform for students and university staff. It provides a motivation for developers all over the world to use and expand the source code. OpenUSS currently only supports HTML browser access. Thus, I had the idea and motivation to design and implement a mobile access solution for OpenUSS using WAP/WML. This will enable OpenUSS subscribers to send and retrieve information whilst on the move, by means of mobile handsets. Chapter 1 will provide more details about Open Source and OpenUSS.

Today's technology of mobile devices should be able to retrieve such information using the wireless application protocol. Chapter 2 illustrates the wireless protocol and the existing mobile technology such as CSD (circuit-switched data), GPRS (General Packet Radio Service), and future technologies such as UMTS (Universal Mobile Telephone System) as bearer services in order to access the WAP/WML interface of OpenUSS.

This project deals with the implementation of WAP as an extension to the OpenUSS system using WML and Java. In addition, current technologies such as i-mode and Java2MicroEdition (J2ME) are evaluated. Chapter 2 and 3 represent the features of those technologies as well as the advantages, the limitations and the reason why WAP is preferred for this project. Moreover, the software development of OpenUSS using the technology of Java2 Enterprise Edition (J2EE) is explained.

A software engineering approach using methodology such as Unified Modelling Languages (UML) is used as a discipline when software is designed and built. Chapter 4 deals with the development process, the problems and their solution, and the evaluation of the software.

1 Open Source Software and Open University Support System

1.1 Open Source Software

Generally, there are two types of software, namely free software and proprietary software. As opposed to proprietary software, free software does not forbid duplicates, changes, and expansion. The basic idea behind the open source is simply: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves through improvement, adaptation, and fixing bugs¹.

Richard Stallman is the initiator of open source as free software when he formed the Free Software Foundation and its GNU Project. GNU is a recursive acronym for "GNU's Not Unix"; it is pronounced "guh-NEW"². The Open Source definition was a policy document of the Debian GNU/Linux Distribution. Debian is an early Linux system that was built entirely of free software and is still popular today. The definition of free or non-free software was written by Bruce Perens and documented as "The

¹ <http://www.opensource.org>

² <http://www.gnu.org>

Debian's Free Software Guidelines", which is refined into the "Open Source Definition".

The Open Source definition lists certain rights that a software licence must be granted in order to be Open Source. Therefore, there are licence models of open source software, which have the aim to protect the software owner from any liability related with the program since the source code is usually available at no charge. The licence models of open source software are as follows³:

- **The GNU General Public License (GPL) versus the Lesser General Public License (LGPL)**

Anyone can change the source code but the GPL does not allow people to make modifications privately. It means that the modifications made must be distributed under the GPL. Thus, it is likely that the author of a GPL-ed program receives improvements from others, who modify their software for their own purposes.

The GPL doesn't allow the incorporation of a GPL-ed program into a proprietary program. The GPL's definition of a proprietary program is any program with a license that does not hold as many rights as the GPL. This is not the case with LGPL, which allows for integration with almost any kind of software, including proprietary software.

An example of a GPL/LGPL case is the Linux-Kernel user interface Kool Desktop Environment (KDE). The authors distributed their programs with Trolltech's product names Qt before an Open Source license was placed on Qt. However, this right does not extend to any third parties that redistribute the program - they must follow all of the terms of the license, even the ones that the copyright holder violates, and thus it was problematic to redistribute a GPL-ed program containing Qt. The KDE developers solved this problem by applying the LGPL, rather than the GPL, to their software.

- **The BSD (Berkeley System Distribution) and X License**

The BSD license is different from the GPL and LGPL. Anyone can change the source code and the modification made must not be given as an open source. However, this license requires programmers to mention in a footnote that the software was developed at the University of California any time a feature of a BSD-licensed program in advertising is mentioned.

In contrast, X-licensed modification can be made privately. In addition, one can sell the binary versions of the program without distributing the modified source code, and without applying the X license to those modifications. This is still Open Source, however, as the Open Source Definition does not require that modifications always carry the original license. As opposed to BSD, the X license must not to be applied to those modifications and one does not have to make provision for advertising.

- **The Netscape Public License (NPL) and the Mozilla Public License (MPL)**

Netscape developed the Netscape Public License (NPL) when the product Netscape Navigator Open Source was made. In fact, the Open-Source version is called Mozilla and Netscape reserves the trademark Navigator. To address this concern, Netscape created Mozilla Public License (MPL). The MPL is much like the NPL, but does not contain the clause that allows Netscape to re-license any modifications. Like X licence, NPL and MPL allow people to make modifications privately.

³ <http://www.perens.com/Articles/OSD.html>

Table 1 shows an overview of the licence models of open source software and their condition:

License	Can be mixed with non-free software	Modifications can be <i>taken privately</i> and not returned to you	Can be re-licensed by anyone	Contains special privileges for the original copyright holder over your modifications
GPL	no	No	no	no
LGPL	yes	No	no	no
BSD	yes	Yes	no	no
NPL	yes	Yes	no	yes
MPL	yes	Yes	no	no

Table 1

1.2 Open University Support System

Open University Support System (OpenUSS) is an Open Source e-learning platform for students and university staff. The Open Source licence models of OpenUSS are GNU General Public Licenced (GPL) and Lesser General Public License (LGPL) for its components. OpenUSS was developed in the year 2000/2001 and is supported by CampusSource, an Opensource-Initiative of the Ministry for Education, Science, and Research in Germany. CampusSource should act as an information system centre (portal) that supports a further development of virtual e-learning in universities and schools.

The initiator of OpenUSS is Prof. Dr. Lothar Grob, the head of the Institute of Business Informatics and Controlling at the University of Muenster in Germany. Lofi Dewanto, who is carrying out his research at that institute, is the Developer of OpenUSS. OpenUSS has become increasingly popular and is already used by many universities in the world, such as the University of Muenster, the University of Cologne in Germany, Ansted University in Malaysia and Perbanas University in Indonesia.

OpenUSS provides modern application services of information and communication based on the Internet, such as managing personal learning, event-related email distribution, a discussion forum specified by subjects, chat rooms, and an archive system for lecture materials to support offline learning. Students are provided with tailor-made information, by which they can subscribe to all the faculties and subjects they are registered for in a semester. Lecturers or university staff can organise lecture materials and save them in a database so that the information for both the past and present time can always be provided. New lecture material can be published via upload and download directly from the lecturer's desktop. All that is required is Internet access and a standard browser. With this basic configuration, any forms of lecture materials such as slides, text documents and spreadsheets can be published and distributed swiftly. At the end of a semester, all information is archived automatically.

1.2.1 OpenUSS as an Open Source Learning Management System

As an Open Source product, other developers can take advantage of the existing code and use, modify or develop further. Currently, OpenUSS only supports HTML browser

access. The bottom line of success of computer supported universities, however, is mobility. Therefore there is a need to support communication technologies such as the Wireless Application Protocol. In this way, students and university staff can retrieve information irrespective of their location and from computers at universities, at home, or via mobile devices.

WAP is already used by many universities in the United Kingdom, such as the University of Southampton⁴, University of Middlesex, and Bath Spa University College⁵. The aims of the services range from access to the university's library catalogue to alumni service.

The mission of OpenUSS is to establish a standard Open University Support System Application Programming Interface (API)-System that can be used for all universities and faculties⁶. The portal should be a communication centre for all members of the organisation.

The OpenUSS concept is based on the ASP (Application Service Provider) model, which means that one or more organisations such as universities, schools, communities and companies can be handled within one instance/installation. In this way, OpenUSS gives users the flexibility to use their chosen devices – the so-called multi-channel information delivery – to access the instance⁷. Figure 1 shows the concept behind OpenUSS:

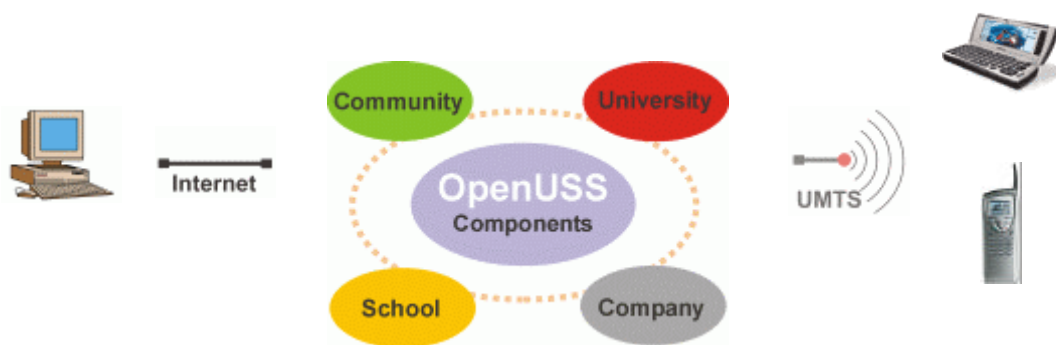


Figure 1

The components of OpenUSS are divided into two parts: foundation component and extension component layer. The components of the foundation layer are essential within OpenUSS. They represent domain-oriented components such as⁸:

- *Student*. This component cover all properties and activities of a student such as name and e-mail address.
- *Assistant*. Component such as lecturers, assistants, and secretaries who are specified by name and title.
- *Administrator*. He or she who manage the system of OpenUSS, such as creating a faculty or institute and allocating rights for accessing the system.

⁴ http://www.mech.soton.ac.uk/edmc/In_The_Media/in_the_media.html

⁵ <http://www.cs.waikato.ac.nz/~mattj/wap.htm>

⁶ Developer's Manual OpenUSS

⁷ <http://www.openuss.sourceforge.net>

⁸ Developer's Manual OpenUSS

- *Faculty*. This component deals with the faculties in a university.
- *Semester*. A component that assigns the season for each faculty. This could be winter and summer semesters for example.
- *Subject*. This component covers the subject within the faculty in the given semester.
- *Security*. Component that deals with the login for assistants and students.

All the functionality of OpenUSS is implemented as Extension Components. The components in this layer are⁹:

- *Lecture*. That is the component that deals with publishing the lecture materials by the lecturer so that the registered students can download them easily.
- *Exercise*. This component should enable student to download the exercises and assistant to upload, correct the exercises, and submit the marks for them.
- *Mailing list*. Component for students.
- *Discussion* as well as *chat* component. While the chat component is synchronous, discussion is asynchronous.
- *Archives*. This component produces an archive for example for each semester.
- *Virtual assistant*. Component that helps students to organise their timetable. It also enables assistants to write personalised comments of student's performance.
- *Data Warehouse and Data Mining*. This component produces statistical data for assistants and can be used as a system report.

The separation of both the layers should make it easier to develop more functions for the system. Figure 2 shows the architecture of OpenUSS components:

⁹ Developer's Manual OpenUSS

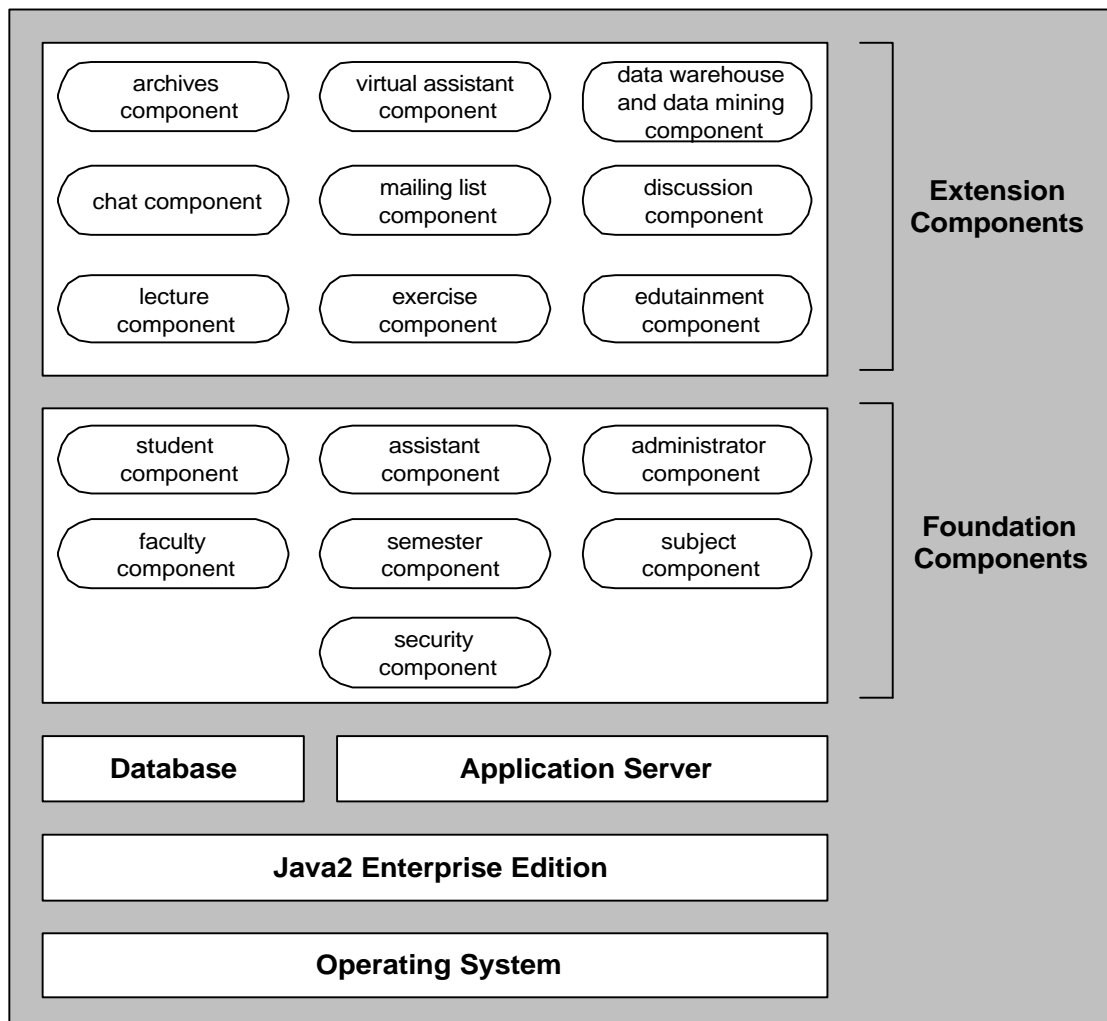


Figure 2

1.2.2 Technical Background of OpenUSS

OpenUSS is a fully-fledged Java-Application based on Enterprise JavaBeans (EJB) and it uses an Application-Server-Technology to enable the processing of a huge application. OpenUSS is built based on the Multi Tier Architecture of Java2 Enterprise Edition (J2EE), which is separated into Presentation Layer, Business Process Layer and Data Layer. These layers are the most important part of OpenUSS technical design. The Presentation Layer is implemented with a Servlet API, which is only HTML at present. The Business Process Layer is designed and implemented with Enterprise JavaBeans (EJB). Any database system, which supports Java Database Connection (JDBC) can be used to cover the Data Layer of OpenUSS¹⁰.

Figure 3 illustrates the technology used by OpenUSS.

¹⁰ <http://www.openuss.sourceforge.net>

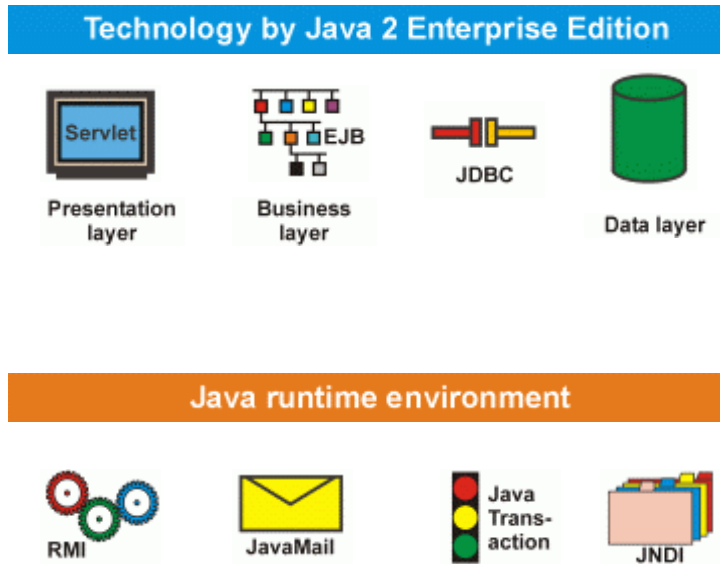


Figure 3

This project deals only with the presentation layer for mobile access of the OpenUSS. The details of J2EE and the development of the presentation layer using WML and Java servlets are shown in Chapter 3 and 4. Before going into the details, a background of current mobile technologies including bearer services as well as the presentation techniques will be illustrated.

2 Wireless Application Protocol and Wireless Markup Language

2.1 Introduction to Wireless Application Protocol

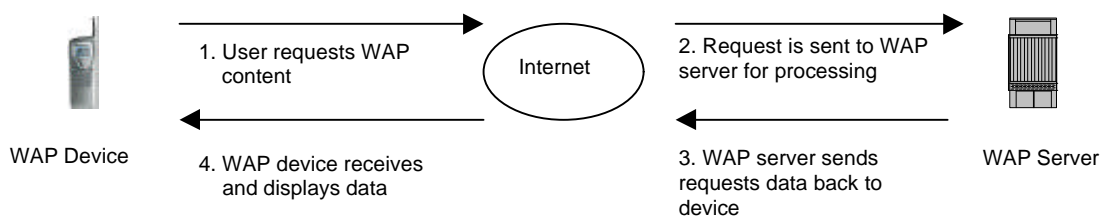
Wireless Application Protocol (WAP) is the de facto standard for wireless computing managed by a consortium of vendors called WAP Forum. The purpose of WAP is to enable easy, fast delivery of relevant information and services to mobile users despite restrictions such as small screens and limited keyboards. WAP is a collection of protocols and standards that enable browser type applications to run in constrained devices over low bandwidth/high latency networks. The complete protocol stack of WAP can be seen in the table below¹¹:

<i>Abbreviation</i>	<i>Name Description</i>
WAE	Wireless Application Environment is an application layer, which includes the micro-browser on the device, WML (the Wireless Markup Language), WMLScript (a client-side scripting language), telephony services, and a set of formats for commonly used data such as images.
WSP	Wireless Session Protocol is a session layer, providing HTTP 1.1 functionality, with basic session state management, and a facility for reliable and unreliable data push and pull.

¹¹ Forta et al, 2000

WTP	Wireless Transaction Protocol is a transaction layer that provides transport services (one way and two way), and related technologies.
WTLS	Wireless Transport Layer Security is a security layer, providing data security, privacy, and authentication, as well as protection against denial-of-service attacks
WDP	Wireless Datagram Protocol as a general transport layer.

WAP content can be served by installing a WAP server. This is a software that behaves like an HTTP server and can be run on the same machine. As can be seen in Figure 4 below, the WAP device makes a request to the WAP server that returns the requested data to the device for processing.



WAP devices request and receive data from WAP servers.

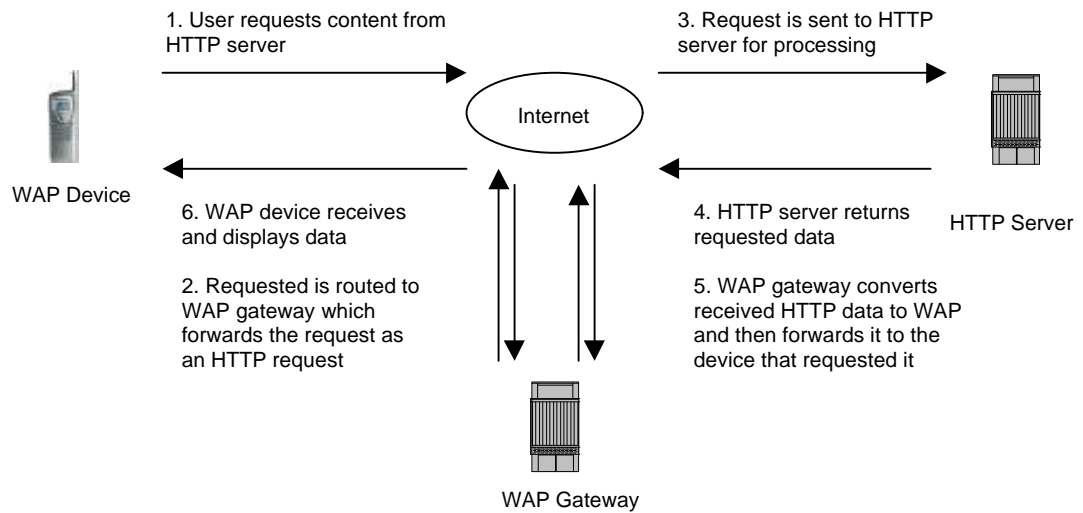
Figure 4

Wireless Session Protocol (WSP) specification defines the WSP push operation and a WSP push Protocol Data Unit (PDU). A push operation is not specified for the HTTP protocol. It is used by the WAP proxy server (often called WAP gateway) to communicate with content hosts. To support pushes, the server has to provide an application interface to allow server based applications to generate a push to a mobile client. The support of pushes on the client side depends on the capabilities of the handsets to handle pushed content. The Nokia OTA configuration proposal to the WAP Forum describes the use of a connectionless push over the SMS bearer, to transfer the configuration data to the handset¹².

WAP devices can request data from an HTTP server through a WAP Gateway, which is located between the WAP device and the HTTP server. The WAP gateway acts as an interpreter between WAP devices and the HTTP server and it handles all data forwarding and filtering or conversation so that the devices just get WAP and not HTTP. This means that WAP service providers only need to have a web server to be able to serve WAP content. For some providers, it may be desirable to have a WAP gateway, and this is easily accommodated. Depending on the configuration access, the content can be granted from any WAP gateway, or only from specific gateways. Figure 5 summarises how a WAP gateway works¹³:

¹² <http://www.handytel.com/technology/wap08.htm>

¹³ Forte, et al, 2000



WAP devices can request and receive data from HTTP servers via WAP gateways.

Figure 5

The first version of WAP (version 1.0) was released in 1998. Since then, there have been enhanced versions such as WAP version 1.1, version 1.2, and version 1.3. The latest version of WAP (version 2.0) has been launched on 31 July 2001 by including new features such as the ability to include new colour, multi media messaging (MMS), large-file downloading, improved navigational menus, and user friendly menus¹⁴.

While WAP V1 uses Wireless Markup Language (WML), WAP V2 uses eXtensible HyperText Markup Language (XHTML) and WML. This means that WAP V2 also provides backwards compatibility to the existing WAP content and thus protects the investment in the previous version. In the use of the WAP V1 protocols (WDP, WTP and WSP), WDP uses UDP/IP over all bearers that support IP access, e.g. GSM-GPRS, and standard LANs etc. In addition to this WAP stack introduced in WAP V1, WAP V2 adds support and services on a stack based on the common Internet stack namely TCP/IP, Transport Layer Security (TLS) and HTTP. These added features in WAP V2 adopt the most recent Internet standards and protocol.

For this project, I have implemented Wireless Markup Language (WML), namely WAP version 1.1., and have tested the application using a simulator from Openwave SDK 5.1. Although this simulator includes a phone simulator incorporating Openwave Mobile Browser 6.0, which supports the WAP 2.0 standard, WAP 1.1 is preferred, since XHTML phones will be expected in late 2002 or early 2003 in Europe¹⁵. Nevertheless, as mentioned above, the full backwards compatibility support for WML 1 applications is provided in the Wireless Application environment for WAP 2.0. through either native support for both languages WML1 and XHTML Mobile Profile markup language (XHTMLMP) or by a defined transformation operation of WML1 to WML version 2¹⁶.

WAP is obviously very simple and graphics-free in a similar way to the early days of the web. However, it is an effective communication medium and can provide simple, fast access to content.

¹⁴ WAP-Forum. Releases WAP 2.0 Specification For Public Review, 1st August 2001

¹⁵ <http://www.openwave.com>

¹⁶ WAP 2.0 Technical White Paper

2.2 Security in the Wireless Application Protocol

Wireless devices make a WAP request through the wireless network to a WAP gateway. The WAP gateway translates requests from a WML browser into HTTP requests for data over the Internet. The WAP gateway executes the request on behalf of the browser, constructing and passing along HTTP variables. Once the Web server processes the request, it replies to the WAP gateway. The HTTP response from the Web application gives XML and the WAP gateway compiles it into the WAP binary XML Content Format (WBXML) and sends the WBXML version of the response to the wireless device¹⁷.

This WAP request/response data path is exactly the same as the request/response cycle to a Web application. In fact, since the WAP gateway is a proxy¹⁸ for the WML browser, the security of the request/response chain has all the features and problems of a secured HTML site. The Wireless Transport Layer Security (WTLS) that belongs to the WAP protocol stack takes the responsibility of the security of this communication. WTLS provides security for the data exchanged directly between the wireless device and the WAP gateway. It works in a very similar way to Secure Sockets Layer (SSL), which ensures confidentiality using public-key cryptography. The gateway translates WTLS messages into SSL and vice versa. Since the WAP gateway is the man-in-the-middle all the conversations between wireless devices and Web applications would be insecure. To accommodate this problem, gateway providers must practice standard security procedures such as¹⁹:

- Employing a firewall.
- Limiting administrative access to the machine to critical parties only.
- Limiting physical access to the machine.
- Only using gateway software that avoids persistent storage of plain text messages; in fact, using software that destroys the plain text as soon as possible.
- Automatically monitoring the machine for new process creation and other indications of compromise with professional management software.

A secure WAP gateway will provide an excellent confidentiality and data integrity of a platform.

2.3 The Wireless Markup Language

The markup language used for describing the structure of documents to be delivered to wireless devices is called Wireless Markup Language (WML). As a desktop computer uses the HTML Internet browser, WML is used by wireless browser. WML was created to deal with the display, bandwidth, and memory limitations of mobile and wireless devices such as cellular phones. WML was based on eXtensible Markup Language (XML) so that the language would survive the demands and fluctuations of turbulent

¹⁷ Forte et al, 2000

¹⁸ **Proxy** is an interface-specific object that provides the parameter marshaling and communication required for a client to call an application object that is running in a different execution environment, such as on a different thread or in another process. The proxy is located with the client and communicates with a corresponding stub that is located with the application object that is being called.<http://www.host-web.fr/iishelp/adc/docs/adcdef01_1.htm>

¹⁹ Forte et al, 2000

standardisation. By using XML as a base, WML was designed to be a lightweight protocol that would meet bandwidth limitations of existing mobile devices. The WML supports following areas such as²⁰:

- *Text presentation and layout* – Line breaks, text formatting, and alignments are supported by WML, although specific devices and WML browsers vary in their output of WML code.
- *Images* – WML supports the Wireless Bitmap (WBMP) image format and image alignment on the screen. An example of using the WBMP for this project is illustrated in Chapter 4.4.
- *User input* – WML support choice lists, multilevel choice lists, text entry, and task controls.
- *Card and deck organisation* – While a single HTML web page viewed is essentially the contents of a single .htm file, a WAP 'page' sends more than one of these pages at once. This is done because of the small screen sizes, so that the phone can cache multiple fragments of content for future use. This collection of fragments is known as a 'deck', and the page fragments themselves are known as 'cards'. Each deck can also feature a 'template' which defines common characteristics for each of the cards within that deck. The multiple WML cards in one deck will be saved as a single file. An example of implementing card and deck for this project is illustrated in Chapter 4.4.

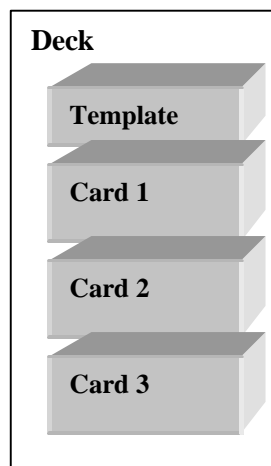


Figure 6

- *Navigation* – WAP supports the standard Internet URL naming scheme and anchored links, allowing navigation between cards in a deck, between decks, or between other resources on the network.

2.4 WAP as an Independent Protocol and the Bearer Services

WAP is important because it provides an advanced path for application developers and network operators to offer their services on different network types, bearers and terminal capabilities. The design of the WAP standard separates the application elements from the bearer being used. This helps in the migration of some applications from Short Message Service (SMS) or Circuit Switched Data (CSD) to General Packet Radio Service (GPRS).

²⁰ Forte, et al, 2000 and Pearce, 2000

WAP is designed to work with most wireless networks such as Code Division Multiple Access (CDMA), Global System for Mobiles (GSM), or Universal Mobile Telephone System (UMTS). Furthermore, WAP has been designed to work with all cellular standards. It can be built on any operating system including PalmOS, EPOC, Windows CE, OS/9, and JavaOS, etc²¹.

2.4.1 Short Message Service

Short Message Service (SMS) with its limited length of 160 characters per message is not normally an adequate bearer for WAP. The overhead of the WAP protocol would be required to be transmitted in an SMS message, which means that even for the simplest of transactions several SMS messages might have to be sent. This involves time and is cost consuming²².

2.4.2 Circuit-Switched Data Transmission Services

Many older WAP based services use CSD as the underlying bearer. CSD is a data transmission service that requires the establishment of a circuit-switched connection before data can be transferred from source data terminal equipment (DTE) to a sink²³ DTE using a connection-oriented network. The so-called dial up connection can take about 10 seconds to connect the WAP client to the WAP Gateway. For WAP phones that do not support V.110 the digital protocol, or the WAP Gateway that does not have a digital direct connection such as ISDN into the mobile network, there is a lack of immediacy and the connect time could increase to about 30 seconds. V.110 is the digital protocol that means that an end-to-end digital call can be made without the need for modem handshaking providing that the network operator has a digital private wire between the WAP Gateway and the mobile network. V.110 cuts the connection time for end to end digital calls to 4-8 seconds typically²⁴. Without V.110, it can be expensive to establish frequent data connections with minimal data transfer volume.

2.4.3 General Packet Radio Service

General Packet Radio Service (GPRS) provides a solution to the problem of long connection set-up times by introducing new nodes into the GSM (Global System for Mobile communications) network to allow packet switched data traffic. This bearer has no dial-up connection. GPRS offers faster data transmission via a GSM network within a range 9.6 kbps to 115 kbps. The signalling and data traffic do not travel through the GSM network. The GSM network is only used for table look up, in the Location Register (HLR and VLR) databases, to obtain GPRS user profile data²⁵.

GPRS infrastructure and mobile phones support data transmission speed of up to 13.4 kbps per channel. Because more than one channel is used for downlinks, GPRS mobile

²¹ Open Mobile Alliance Ltd, 2002

²² <http://www.handytel.com/technology/wap07.htm>

²³ Data Terminal Equipment (DTE): In a data communication network, the data source, such as a computer, and the data sink, such as an optical storage device. Glossary can be found at <http://www.provu.co.uk/glossary.html>

²⁴ <http://www.handytel.com>

²⁵ <http://www.cellular-news.com>

phones allow for higher data transmission speeds. There are several types of phones, which use a different number of channels for data transmission¹⁸:

- Type 2+1 – two downlink channels and one uplink data transmission channel that can receive 26.8 kbps and send 13.4 kbps.
- Type 3+1 – three downlink channels and one uplink data transmission channel that can receive 40.2 kbps and send 13.4 kbps.
- Type 4+1 – four downlink channels and one uplink data transmission channel that can receive 53.6 kbps and send 13.4 kbps.

The GPRS mobile phones are classified into the following three classes in terms of the possibility of simultaneous calls (via GSM) and data transmission (via GPRS)¹⁸:

- Class A – Simultaneous calls (via GSM) and data transmission (via GPRS)
- Class B – Automatic switching between the GSM and the GPRS mode is possible according to telephone settings.
- Class C – Hand operated switching between the GSM and the GPRS mode

With GPRS, any channel that is not busy with a call is pooled into one packet channel, which is shared among all users that want to send and receive data. When more users make calls in the cell, the available bandwidth for data traffic decreases, and when they hang up, it increases.

One efficient way of sending content to a mobile phone is by the user, maintaining more or less a permanent GPRS (mobile originated) session with the content server. However, mobile terminated IP traffic might allow unsolicited information to reach the terminal and this may cause Internet sources to be not chargeable. A possible worse case scenario would be that mobile users would have to pay for receiving unsolicited junk content. It means that by originating the session themselves from their handset, users confirm their agreement to pay for the delivery of content from that service. However, users could make their requests via a WAP session, which would not therefore need to be blocked. As such, a WAP session initiated from the WAP microbrowser could well be the only way that GPRS users can receive information onto their mobile terminals²⁶.

Because the bearer layer is separated from the application layer in the WAP protocol stack, WAP provides the ideal defined and standardised means to port the same application to different bearers. As such, many application developers will use WAP to facilitate the migration of their applications across bearers, once GPRS based WAP protocols are supported.

2.4.4 Universal Mobile Telephone System

UMTS stands for 'Universal Mobile Telecommunications System' and is one of the major new 'third generation' (3G) mobile communications systems already launched. In contrast to the GSM-Standard, the net structure and the method of the radio system of UMTS can be developed separately and they can still match each other. The important reason for this is because UMTS is a group of standard that can be employed

²⁶ <http://www.handytel.com>

alternatively since there is no worldwide standard. The method of the radio system is the CodeDivisionMultipleAccess2000MC-standard that set up the American CDMA net and WidebandCDMA. WCDMA offers Frequency Division Duplex (FDD) and Time Division Duplex (TDD). Europe and Japan deploy predominantly WCDMA-FDD and use WDMA-TDD just in special cases²⁷.

According to the UMTS forum, UMTS will play a key role in creating a mass market for high-quality wireless multimedia communications that will exceed 2 billion users worldwide by the year 2010 and over 100 3G licenses have already been awarded²⁸.

The licence fee of UMTS was very high (63 billion (bn) Deutsche Marks (\$29.29bn) in Germany, and in the UK £22.5bn (\$34.5bn)) but the forecasts for Germany's auction were halved after the licences covering the Netherlands raised \$2.4bn, a third of the amount expected. That uncertainty has led to analysts predicting that the total raised by the licences could end up anywhere between 25 and 61 billion Euro²⁹. Nevertheless, the UMTS forum said that UMTS will deliver low-cost, high-capacity mobile communications, offering data rates as high as 2Mbit/sec, which will be available in certain small areas if a user is standing still and using the base station alone. A user driving a car in rural areas will probably not be able to use more than about 100 kbps.

Although UMTS can deliver high-value broadband information, as well as commerce and entertainment services to mobile users via fixed, wireless and satellite networks, UMTS will provide data traffic rather than voice services because of the high cost. With the latest WAP version 2.0 and the higher capable network bearers such as GPRS and UMTS, new types of content such as steaming media and provide ,always on' availability are permitted.

Finally, the bandwidth required by application users can be expected to steadily increase. Therefore, there is still a need to optimise the device and network resources for wireless environments. Multimedia applications enhancement in WAP version 2.0 is the most relevant step. If WAP is very successful in mass-markets on 2.5G networks, 3G networks may be needed purely for capacity relief.

2.5 Presentation Techniques: i-mode versus WAP

I-Mode is a product from the Japanese operator telco NTT DoCoMo. While WAP uses WML, i-mode services are written in cHTML. It is said that this is an advantage as cHTML is only a subset of HTML and it is easy for interface programmers to adapt. However, WML is very similar to HTML too and it is doubtful that HTML programmers would have difficulties in adapting to it.

In addition, i-mode has focused on adapting terminals to the type of content served, while WAP has adapted the content to the terminals that should handle it. The outcome of this difference is that i-mode is very graphic and appealing, while WAP has a very flexible and effective protocol stack³⁰. NTT Docomo has already launched i-mode in Europe in June 2002 and already has over 34,000 i-mode customers in Germany (E-Plus) and the Netherlands. To stay online on the network, E-Plus uses the packet

²⁷ Glahn, 2001

²⁸ The lists of countries, which have been awarded can be found at
<<http://www.umts-forum.org/licensing.html>>

²⁹ <http://news.bbc.co.uk/1/hi/business/876554.stm>

³⁰ WML versus cHTML <<http://www.netlight.se/imodevswap.html>>

switching technology GPRS³¹, which makes access faster, and allows users to pay only for the amount of data downloaded rather than for the duration of the connection.

According to the experts, the expected launch of Japan's wireless web technology, i-mode, into the European market will not spell the end for Wap applications. Instead, industry players believe that i-mode will merge with WAP rather than replace it. Iveca Juresa, European business manager for mobile e-services at Hewlett Packard said that i-mode will not replace WAP because there are over 400 companies behind the WAP standard today, while i-mode is a standard by one company, NTT-DoCom³².

The use of circuit switched networks today, however, is not due to WAP. WAP works well in packet switched networks. The reason for the circuit switched technology is that WAP today is used in GSM networks. With the introduction of GPRS and 3G networks, this will change.

3 Java 2 Micro Edition, Java 2 Enterprise Edition and the Products

3.1 Java 2 Micro Edition (J2ME)

J2ME, Java 2 Micro Edition is a new, very small application environment with tools and supplies for developing applications for mobile devices. It targets mobile devices with the runtime of an equivalent size to WAP 2.0 and i-mode 3.0 browser stacks. It contains:

- Java virtual machines that fit inside the range of consumer devices.
- A library of APIs that are specialised for each type of device and tools for deployment and device configuration.
- A profile, i.e. a specification of the minimum set of APIs useful for a particular kind of consumer device (set-top, screen phone, wireless, car, and digital assistant) and a specification of the Java virtual machine functions required to support those APIs.

J2ME is the smallest of the JAVA continuum and can be shown in the Figure 7 below:

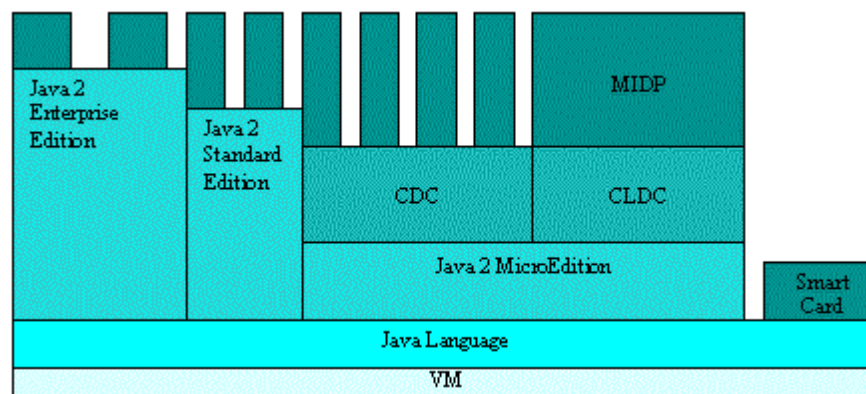


Figure 7

³¹ <http://www.eplus-imode.de/1/de/html/pub/presse/index.html>

³² Woffende, 2000

The configuration consists of a virtual machine, core libraries, classes and APIs. There are two J2ME configurations at present: the Connected Limited Device Configuration (CLDC) and the Connected Device Configuration (CDC). CLDC is designed for devices with constrained Central Processor Unit (CPU) and memory resources. These devices run typically on either a 16- or 32-bit CPU and have 512 Kbytes or less memory available for the Java platform and applications. CDC is designed for next-generation devices that run on a 32-bit CPU and have 2Mbytes or more memory available for the Java platform and applications.

The Mobile Information Device Profile (MIDP) is a set of Java APIs that, together with CLDC will provide a complete J2ME application runtime environment targeted at mobile information devices such as cellular phones and two-way pagers. The MIDP specification addresses issues such as user interface, persistence storage, networking, and application model.

The Pros and Cons are illustrated to compare J2ME to WAP:

- With J2ME it is possible to write device independent applications with more features than for WAP. The applications are easy to download and possible applications are limited almost only by the mind. For example a group of people could have the same address book available on a server somewhere or a photo collection. People could play multi-user games over their GPRS connection or get the latest stock information. Signatures could be written from a distance and payments can be made.
- J2ME needs no gateways. This makes testing simpler, quicker in terms of time and lower in terms of cost.
- While one can just browse with WAP, Java provides interactivity and allows operation offline; For example, playing games.
- The existing HTML sites/solutions must be rewritten. This is the same as WAP.
- The Graphical User Interface components are extremely limited, just for simple application and simple games.
- In terms of security, J2MEs handset maker has the option NOT to do SSL while, as mentioned above in the Chapter 2.2, a security hole can occur in WAP. However, this drawback in terms of security of WAP can be avoided by using technologies such as Firewall etc.

Although J2ME uses a full-features Java-based application environment that provides users with all the varieties of monochrome interface seen on WAP phones today, I have chosen to use the technology of WML/WAP for this project because of the difference in recent development between J2ME and WAP devices. The research shows that there are only 34 mobile phones that support Java devices³³ while 99% of mobile phones on the market support WAP with more than 300 million subscribers³⁴. In addition, the recent development in WAP 2.0 will support the features provided in J2ME. Furthermore, WAP 2.0 supports backwards compatibility, so that a migration of the WAP version used for this project, to WAP 2.0 is possible.

³³ <http://www.javamobiles.com/>

³⁴ WAP Forum Ltd, M-Commerce World, London 2001

3.2 Java™ 2 Platform, Enterprise Edition Specification

Enterprises nowadays gain competitive advantage by quickly developing and deploying custom applications that provide unique business services. This could be an internal application for employee productivity, or Internet applications for specialised customer or vendor services, or institutes. The keys to success are quick development and deployment. In addition, portability and scalability are also important for long term viability. Enterprise applications must scale from small working prototypes, enterprise-wide services, accessible by tens, hundreds, or even thousands of clients simultaneously.

In 1997, Pountain and Montgomery introduced Multi-Tier-Architecture, which is used as a basis today for enterprise applications³⁵. However, multi-tier applications are hard to design because they require a variety of skill-sets and resources. In today's heterogeneous environment, enterprise applications have to integrate services from a variety of vendors with a diverse set of application models and other standards. Industry experience shows that integrating these resources can take up to 50% of application development time. J2EE defines the standard for developing multi-tier enterprise applications and simplifies enterprise applications by basing them on standardised, modular components, by providing a complete set of services to those components, and by handling many details of application behaviour automatically, without complex programming.

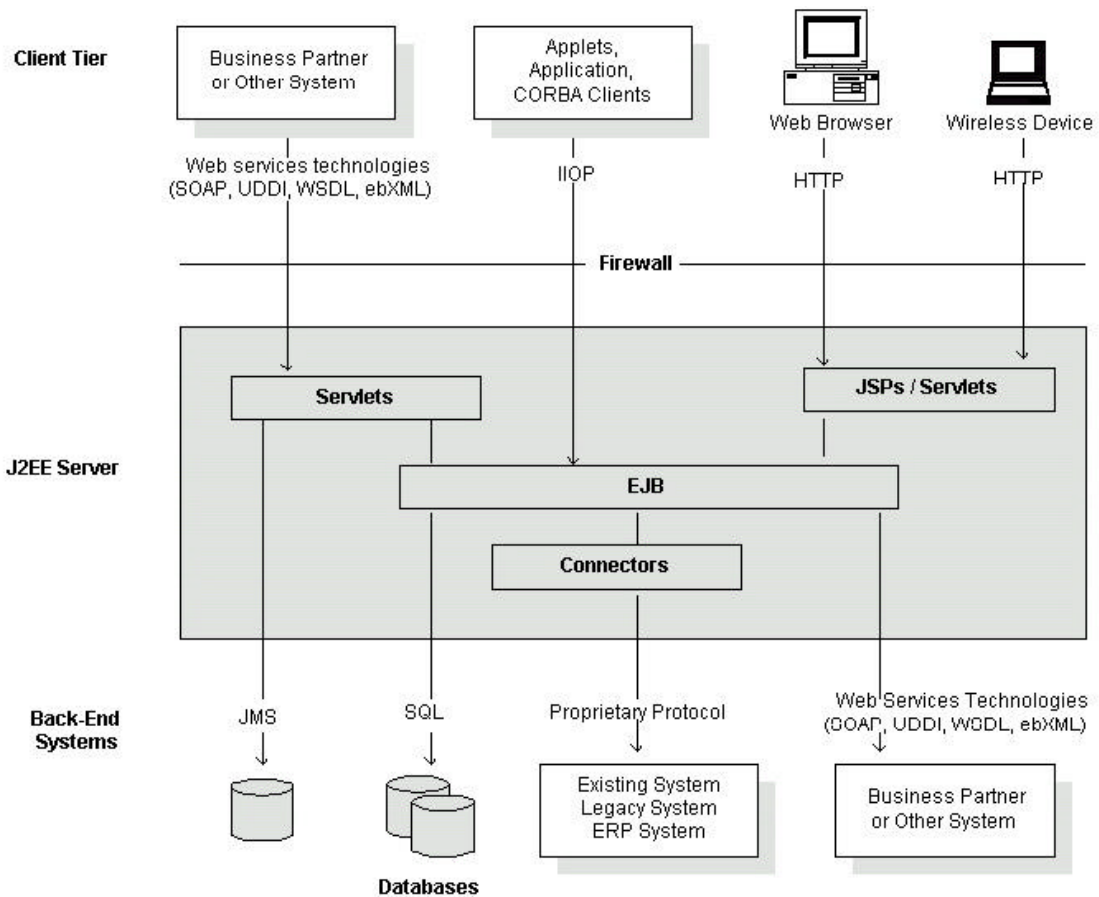
Since J2EE is written in Java, it has "Write Once, Run Anywhere" portability, Java Database Connectivity (JDBC) API for database access, CORBA technology for interaction with existing enterprise resources (Internet Inter-ObjectRequestBroker Protocol (IIOP)-API, and a security model that protects data even in internet applications. Building on this base J2EE adds full support for Enterprise JavaBeans (EJB) components, Java Servlets API, JavaServer Pages (JSP) and XML technology³⁶. The Java Messaging Service (JMS) allows J2EE deployment to communicate using messaging both within and outside the J2EE system. Connectors allow an access of existed information systems from a J2EE deployment. This could be any system such as mainframe systems running high-end transactions, Enterprise Resource Planning (ERP) systems or even proprietary systems. Connectors are useful because they manage the details of middleware navigation to an existing system automatically, for example handling transactions and security concerns. In addition, a single driver can be written to access an existing system to be deployed into any J2EE-compliant server. In addition, this driver can be reused in any J2EE server. This is a huge benefit for independent software vendors (ISVs) who want their software to be accessible from within application servers because they do not have to write a custom driver for each server. J2EE takes Java's Enterprise API's and bundles them together in a complete development platform for enterprise-class *server-side* deployments written in Java.

Figure 8 shows some of the major technologies of J2EE working together³⁷:

³⁵ Pountain, et al, 1997

³⁶ <http://java.sun.com/j2ee/overview.html>

³⁷ Roman et al, 2002



A Java 2 Platform, Enterprise Edition deployment

Figure 8

3.3 Enterprise JavaBeans

The Sun™ Enterprise JavaBeans specification defines architecture and interfaces for developing and deploying distributed Java server applications based on multi-tier architecture. This specification intends to facilitate and normalise the development, deployment and assembling of applicative components. These components are called Enterprise Beans, which will be deployable on EJB platforms. The resulting applications are typically transactional, database-oriented, multi-user, secured, scalable and portable. Precisely, this EJB specification describes two kinds of information³⁸:

1. The runtime environment called EJB server, which provides the execution environment together with the transactional service, the distribution mechanisms, the persistence management and the security.
2. Some kind of programmer's guide and user's guide explaining how an enterprise bean should be developed, deployed, and used.

The EJB™ specification defines a server component model. An Enterprise JavaBean (EB) is a "non visual" software component running on the server-part of an application and may be configured at deployment time by editing its properties. The resources needed by an EB are the API of J2EE the above mentioned, plus transactional services, storage services, security services, naming services, and messaging services. The EJB

³⁸ <http://www.objectweb.org/jonas/current/doc/JOnASWP.html>

server provides such resources to the bean. The interface between an EB and the EJB server is materialised by an architectural component called "**container**". The container is in charge of the EB instances life cycle and persistence, and of the interaction with the transaction and security services. The following parts compose an Enterprise Bean to be developed by the Enterprise Bean Provider³⁴:

- The "Remote Interface" is the client view of the bean. It contains the signatures of all the "business methods" of the bean.
- The "Home Interface" contains the signatures of all the methods for the bean life cycle (creation, suppression) and for instances retrieval (finding one or several beans) used by the client application.
- The bean class, which implements the business methods, and all the methods (described in the EJB specification) allowing the bean to be managed in the EJB server.
- The deployment descriptor, containing the bean properties that may be edited at configuration time.

In the specification, there are three kinds of enterprise beans defined³⁹:

1. **Session beans.** There are stateful and stateless session beans. Stateful beans are objects that retain across client invocations and are not easily pooled and scaled, such as the shopping cart, or transactional data cache State holder (for multi-step form before data is inserted). The opposite of that is stateless. Examples of stateless beans are the workflow engine, catalogue engine, and credit card authoriser.
2. **Entity beans.** Those are objects that represent data in a database. They may be shared by several clients and are identified by means of a primary key. An EJB environment is responsible for managing the persistence of such objects for example a product, an order, an employee, student, stock, or a purchase order.
3. **Message-driven Beans.** They are similar to session beans in that that they are actions. The difference is that message-driven beans are called by sending them JMS (Java Message Service) messages. Examples of this are credit card authoriser, purchase order processor, and order and workflow processor, etc.

There is management and feature of JavaBeans that will be illustrated directly in the next Chapter to gain a better understanding what the further specification of JavaBeans tells and what the chosen JavaBeans product for this project provides the EJB specification.

3.4 The Products of J2EE and the OpenUSS Architecture

As Java2 Enterprise Edition is only a specification, a product of J2EE is needed to enable building such a platform. The products that are used by OpenUSS are JOnAS for the EJB container and Enhydra for the servlets container. Since OpenUSS is Open Source so are JOnAS and Enhydra. JOnAS is a pure Java™ implementation of the EJB™ specification that relies on JDK and is part of the ObjectWeb Open Source initiative. Enhydra is the first and leading Open Source Java/XML application server that was initially created by Lutris Technologies Inc, an Open Source Enterprise

³⁹ Roman, et al, 2000

Software and Services company. The Enhydra.org project is similar to Apache, but with a focus on E-Business software revolving around the application server⁴⁰.

JOnAS implements the EJB specification, by providing all the elements of an EJB Server such as Transaction Manager, Persistence Manager, Security Manager, and the tools to generate containers.

Following are further features of EJB and what the EJB product JOnAs supports⁴¹:

- **TRANSACTION MANAGEMENT.** With EJB, transaction control is no longer hard coded in the server application, but is configured at deployment time. This is known as "*Container-managed transaction demarcation*". With "*Container-managed transaction demarcation*" the transactional behaviour of an enterprise bean is defined at configuration time and is part of the deployment descriptor of the bean. The container is responsible for providing the transaction demarcation for the enterprise beans.

Only two kinds of beans, session beans and message-driven, can be used with "*Bean-managed transaction demarcation*". In this case, the container is responsible to suspend any transaction that may be associated with the client request.

JOnAS is built on top of the Java Transaction Manager (JTM), which implements CORBA Object Transaction Service. The JTM manages distributed transactions, whose context is implicitly propagated with the distributed requests. The JTM may be distributed across one or more EJB servers; thus a transaction may involve several beans located on different EJB servers. JTM's transactions may be demarcated explicitly by the client, or the bean itself. It may also be implicit, i.e. performed by the container according to the transactional attributes values. Figure 9 shows JOnAs's configuration of transaction management of different distribution architecture cases for three Enterprise Beans involved in the same transaction.

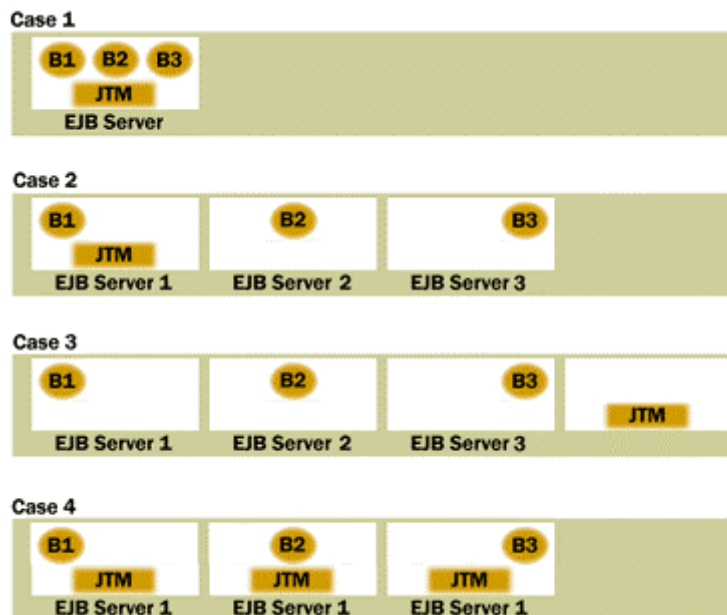


Figure 9

⁴⁰ <http://www.enhydra.org> and <http://objectweb.org>

⁴¹ <http://www.objectweb.org/jonas/current/doc/JOnASWP.html>

Figure 9 allows JOnAS to differ following architecture configurations:

1. Case 1: The three beans B1, B2 and B3 are located on the same EJB server, which embeds a Java Transaction Monitor.
2. Case 2: The three beans are located on different EJB servers, one of them running the Java Transaction Monitor, which manages the global transaction.
3. Case 3: The three beans are located on different EJB servers, the Java Transaction Monitor is running outside of any EJB server.
4. Case 4: The three beans are located on different EJB servers. Each EJB server runs a Java Transaction Monitor. One of the JTM acts as the master monitor, while the two others are slaves.

Of course, the EJB servers may be located on the same or on different machines. Each EJB server runs in a separate Java Virtual Machine (JVM), as well as the JTM when it runs in a stand-alone way (not within an EJB server, as in case 3).

- **PERSISTENT MANAGEMENT.** As mentioned above, the EJB specification defines three kinds of bean: Session beans, message-driven beans, and entity beans. As opposed to session beans, an entity bean represents persistent data. That is an object view of an entity that is stored in a persistent storage such as database. The persistence management of such an object is entirely transparent to the client that will use it, and may be or may not be transparent to the bean provider that will develop it. This conforms to the EJB reference architecture where the client does not see any data access operation, such operations being used on the server side enterprise bean implementation only. An entity bean may be one of the two following categories:
 - An enterprise bean with **Container-Managed Persistence**. In such a case, the bean provider does not develop any data access code, persistence management is delegated to the container. The bean provider describes persistence for the transactional aspect in the bean descriptor. This description provides the mapping between the fields of the bean and the persistent storage (generally the attributes of a relational table of database).
 - Enterprise bean with **Bean-Managed Persistence**. In this case, the bean provider writes the database access operations in the methods of the enterprise bean that are specified for data creation, load, store, retrieval, and remove operations (`ejbCreate`, `ejbLoad`, `ejbStore`, `ejbFind...`, `ejbRemove`).

In both cases, the data access operations are generated by the EJB environment using JDBC or Java Standard Query Language (JSQL) to access relational databases, or using other (proprietary) classes for accessing non relational data storage.

JOnAS supports entity beans that conforms to the EJB specification and is based on JDBC:

- **Bean-Managed Persistence:** the bean provider is able to develop entity beans, the states of which are stored in relational databases. The bean provider should develop the data access methods using the JDBC interface.
- **Container-Managed Persistence:** the EJB environment automatically handles data storage and access operations in a relational database. The bean

provider only needs to provide a bean descriptor containing information about the fields mapping into the database schema and the JDBC DataSource (driver and database URL).

Today, persistence management is also supported via the JDBC interface on most relational databases such as Oracle, Sybase, InstantDb, Postgres, Ingres, Interbase, SQL Server, etc.

- **SECURITY MANAGEMENT.** The aim of the security in the EJB architecture is to control the access to the methods of an EJB. All the concepts of security are based on the notion of **roles**. By a given set of roles the methods can be accessed. In order to access the methods, the user *must* be at least in one of this set of roles. These roles and the mapping between roles and methods (permissions) present a simplified *security view* of the enterprise beans application. The System Administrator has to map the set of security roles to the "specific" roles of a target operational environment, i.e. groups on Unix systems. The EJB specification defines two kinds of security management:
 - declarative security management. It is set by the Bean Deployer or Application Assembler to integrate the security on an EJB component in the target operational environment.
 - programmatic security management. It is used by the Bean Programmer to enforce security in the code of the EJB itself in such situation where the declarative one is not sufficient.

These two kinds of security management are not limited: in most "real world" applications, they complete each other to make the EJB architecture more secure.

In JOnAS, the security is available with the following restrictions:

- User authentication should be provided by the client of the EJB (i.e. the Web server), which must initiate and propagate the security context with calls to EJB.
- There is no mapping concerning the security between JOnAS and the target operational environment. It means that the roles defined for JOnAS cannot be mapped to roles of the target operational environment, i.e. groups on Unix Systems.
- **JAVA MESSAGING SERVICE (JMS).** The JMS API adds to this a common API and provider framework that enables the development of portable, message based applications in Java. Using this JMS API, it is possible for any EJB component to send or "synchronously receive" JMS messages. The bean programmer has the possibility to use resources such as a DataSource. Thus, the bean programmer is able to provide JMS code inside of an EJB method in order to send a message toward a JMS Queue. In addition, JMS API in the J2EE 1.3 platform provide the message-driven bean to enables the asynchronous consumption of messages. A developer can easily add new behaviour to a J2EE application with existing business events by adding a new message-driven bean to operate on specific business events.

For asynchronous EJB method invocation, JOnAS provides Message-driven Beans as specified in the EJB 2.0 specification. A Message-driven Bean is an EJB component which may be considered as a Java Message Service MessageListener, i.e. which processes JMS messages asynchronously: it implements the *onMessage(javax.jms.Message)* method, defined in the *javax.jms.MessageListener*

interface. It is associated with a JMS destination, the *onMessage* method will be activated on the reception of messages sent by a client application to this destination.

- **DISTRIBUTION.** The distributed environment in the EJB world is Remote Method Invocation (RMI). RMI is the Java language’s native way to communicate between distributed objects for example that two different objects running on different machines.

Currently JOnAS is working on two distributed processing environments; RMI using the Sun™ proprietary protocol JRMP and Jeremie, the RMI called Jonathan. With Jeremie, JOnAS benefits from local RMI calls optimisation. In the future, JOnAS will support RMI/IIOP, thus providing CORBA interoperability.

As mentioned above, OpenUSS uses Enhydra as a servlets container. Enhydra is a Java/XML Application Server that multi-tier (presentation, business logic and data store) design and implementation can be distributed across the network for incorporating existing business objects. This configuration can be done as an all-servlet architecture or used in conjunction with an Enterprise JavaBean server for larger mission critical deployments. The Enhydra Java Application Server is functionally complete as a general-purpose application server. But, when the need arises, new components can be added, or existing components can be swapped out and replaced with customer-specific functionality, such as building access control around a proprietary, legacy interface. Once compiled, the Enhydra application server serves applications through standard Web servers, such as Apache or Netscape. Enhydra is also capable of acting as a Web server, handling HTTP requests directly from browsers⁴².

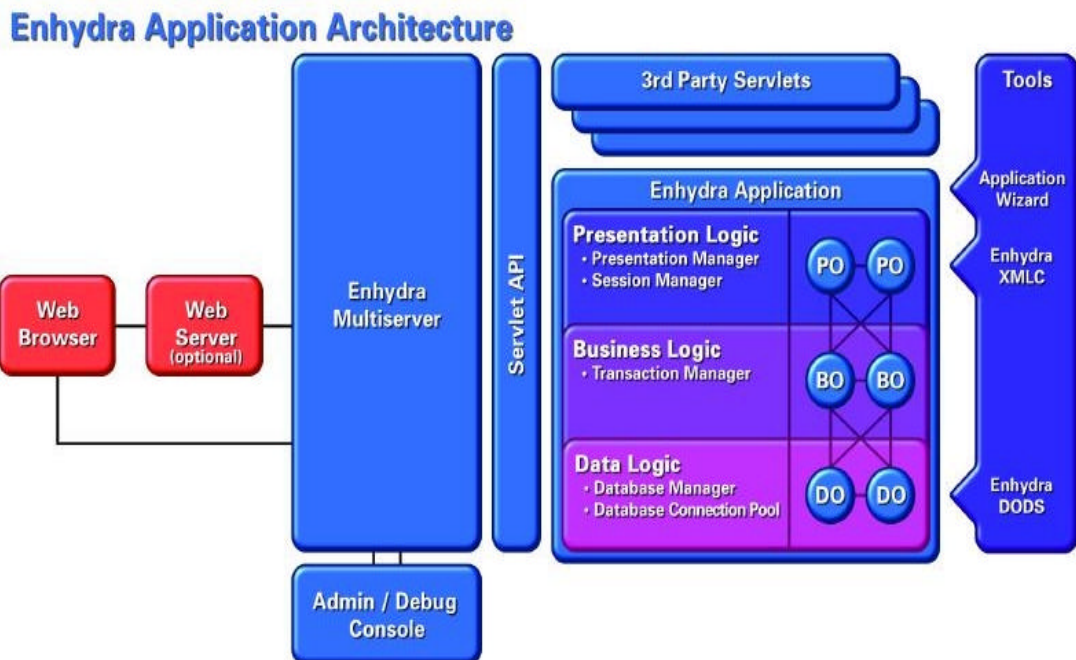


Figure 10

⁴²<http://www.enhydra.org>

Figure 10 shows the Application Architecture of Enhydra. Following is the illustration of run-time modules and tools include⁴³:

- ENHYDRA APPLICATION FRAMEWORK – That is a super-servlet run-time environment of common services (session, presentation, database connectivity) for supporting N-tier Enhydra applications. The Enhydra Application Framework provides all the infrastructure of a Web application, but none of the content. An application object and a set of presentation objects are simply to be added to have a complete Web application.
- ENHYDRA MULTISERVER – Enhydra applications may be run from any Servlet-compatible Web server, such as Apache or Netscape Enterprise Server. They may also be run from the included Enhydra Multiserver, which acts as a stand alone Web server. The Multiserver also provides Common Gateway Interface (CGI) and Web Application Interface (WAI) so that applications can be run from nearly any Web server. The Multiserver efficiently pools and services request.
- ENHYDRA XMLC – XML Compiler designed to support vastly improved designer/developer co-development and create true separation between the presentation and business layers. Enhydra XMLC is an XML compiler that converts document templates, including HTML, cHTML, WML, XHTML, and custom XML, into resources that can be directly accessed from a Java program using standard Document Object Model (DOM) manipulation syntax. DOM allows access to XML files so that the XML Compiler will also allow for presentation objects to serve dynamic content XML. It is used to provide access methods to read and modify the content. An example of this use for this project is illustrated in Chapter 4.4.
- ENHYDRA Java Designer Developer Interface (JDDI)-Compiler – a structured approach to using embedded Java for dynamic HTML. The Enhydra JDDI compiler provides developers and Web interface builders with the ability to construct highly maintainable dynamic HTML presentation objects of integrated HTML and server-side Java. Sections of embedded Java HTML and Java functionality are integrated via easy-to-maintain "sections". Sections make it easy to separate Java areas from the more static HTML sections. This approach to User Interface design gives HTML designers the ability to access to dynamic data generated by Java sections without accidentally corrupting Java code. Access to dynamic data and decoded CGI parameters are provided to both HTML and Java through Enhydra JDDI Fields. Compiled with the Enhydra JDDI compiler into Java classes, applications can be shipped as convenient, high performance jar files⁴⁴.
- ENHYDRA Distributed Oceanographic Data System (DODS) – DODS is a graphical object-oriented design tool using Java Foundation Classes (JFC) that generates the data objects-to-relation database code. DODS allows you to design the data-layer classes for an application that will utilise the Enhydra Application Framework. DODS generates and compiles the resultant Java source code.

⁴³[<http://enhydra.enhydra.org/software/tour/index.html#eaf>] and

[<http://www.pisoftware.com/publications/JavaColumn/appservers.9902.html>]

⁴⁴ The Java™ Archive (JAR) file format enables you to bundle multiple files into a single archive file.

<<http://java.sun.com/docs/books/tutorial/jar/>>

- ENHYDRA APP WIZARD – The application wizard is designed to quickly bootstrap the application development process. The application wizard creates a new source tree for a simple Enhydra Web application. When executed, it asks for the name of the application for which the application environment is to be instantiated. For teaching purposes, the Wizard application is also an excellent example itself of an Enhydra application.
- ENHYDRA MANAGERS – Presentation Manager manages the presentation objects (.po-objects), session manager enables state management within the application that is essential for an application such as a transaction oriented e-Commerce application. The database manager maintains a pool of JDBC connections to a database, allowing for much faster database access.

Enhydra has integrated JOnAS within Enhydra Enterprise and is contributing to JOnAS developments. All in all JOnAS and Enhydra have a good rating as application server. The concept and the architecture of the server are very well elaborated. Designing the application in this way minimises the impact on the application when databases, file formats or URL layouts are switched.

4 OpenUSS-WAP Software Development

4.1 Requirement Analysis and Use Cases

Requirement analysis is a software-engineering task that bridges the gaps between system level requirements engineering and software design⁴⁵. Requirement analysis is important to build a software solution that solves problem.

For this project I have designed a program that enables OpenUSS-System's users to retrieve the exact information that is provided on the OpenUSS HTML web via mobile devices. In the development of requirements several assumptions are made for problem recognition, evaluation and synthesis, and review in software requirement analysis:

- Information such as registration, faculties, subjects, and news have to be provided at the first stage on the HTML web of OpenUSS before data can be retrieved via WAP. That information will be saved in the database, which will be accessed through WAP. Users can only retrieve information via WAP and not create, delete, or download files.
- The system is accessible by two separate user groups, namely students and lecturer. Definition of lecturer is staff member such as lecturer, assistants and other universities staffs, who have an access right. In addition, user can access Faculty in the system directly in order to retrieve news of faculties promptly without login action. However, no news of a subject can be retrieved, since one has to be registered to be able to do that.
- A WAP user interface will offer the possibility for users to select the user group. This of course is only valid for those who are registered and given access to it.
- After selection of the correct role, users authenticate themselves via username and password (login).
- Once users have logged on they can choose from the following options:
 - a **student-role** providing details on
Registered faculties & subjects where the student is enrolled.
 - Information on all subjects during a semester. Semester could be named e.g. summer and winter semester
 - Subject-specific information on courses and new lecture material
 - a **lecturer/staff-role** providing details on
Registered faculties
 - Information on faculties a user is registered with
 - Once the lecturer has selected the registered faculty number of a semester he/she is teaching, a subject list appears. By clicking on this list individual subjects come up. One can select a subject and access details such as news or lecture material. This is in line with the student interface. This will allow a lecturer to ensure that lecture material was put on the web and can be accessed and

⁴⁵ Pressmann, et al, 2000

he/she will also be able to monitor the activity of other lecturers within the same faculty by e.g. reading the news.

- A 'back' button as well as 'go back to main menu' button is provided to simplify the search functionality for users.
- An error message for giving the wrong login is also available.

After requirements were gathered, a set scenario is created to identify a thread of usage for the system to be constructed. The actors that communicate with the system are identified, student and staff. The use cases of the problem domain specification are shown in Figure 11 as follows:

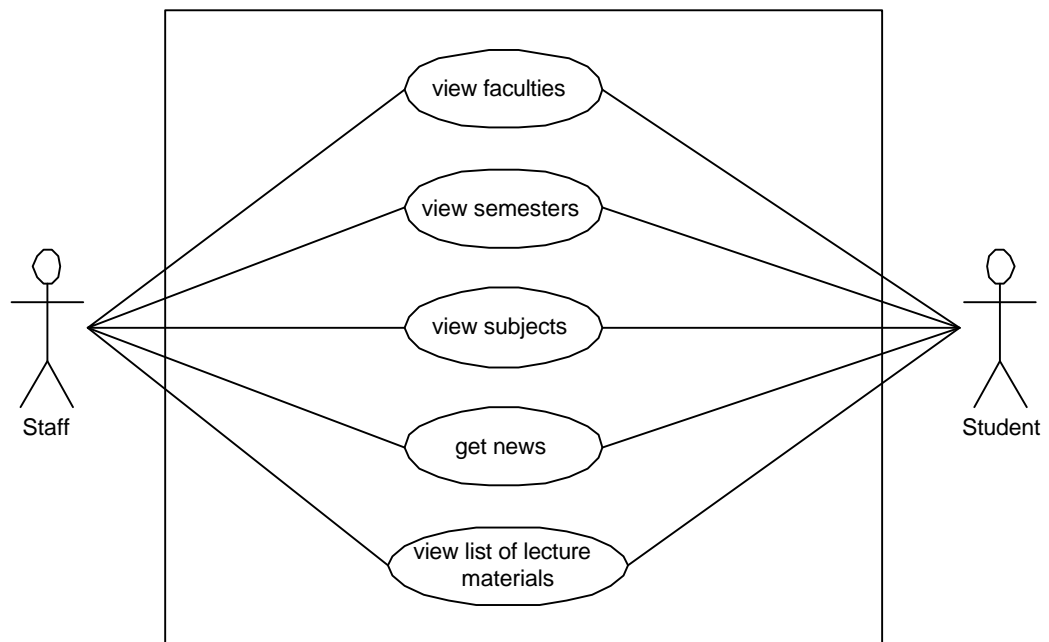


Figure 11

4.2 Activity Diagram

By eliciting information from the developer of OpenUSS and gathering documents, data, functional and behavioural requirements are identified. To understand the information domain, an Activity Diagram is used to describe the *workflows*. There is a back button of each page that goes to the previous page. In addition to back button, depending on the mobile devices, in the login page for lecturers/staffs and students there is also an edit button in case of entering wrong username or/and password. As it is the nature of mobile devices, the WAP-session can be terminated anytime. Figure 12 illustrates the activity diagram:

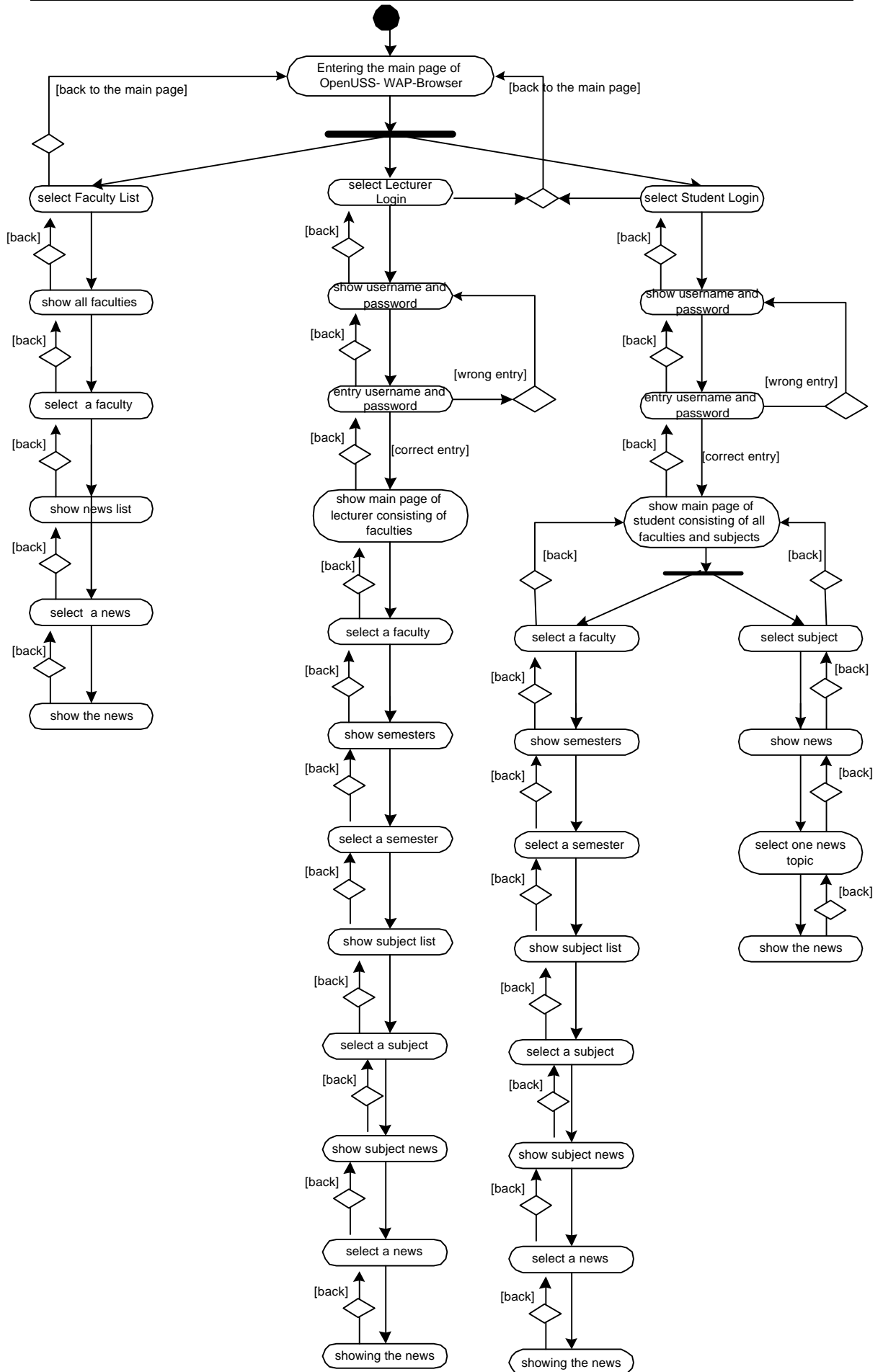


Figure 12

4.3 Class Diagrams

As illustrated in Chapter 1.2.1, OpenUSS consists of two components, Foundation and Pluggable Components. This Project deals only with the foundation component i.e. student, assistant (lecturer), faculty, subject, semester, and enrolment are due to the limited time given. The existing class diagram of the Business Layer of OpenUSS below illustrates these components:

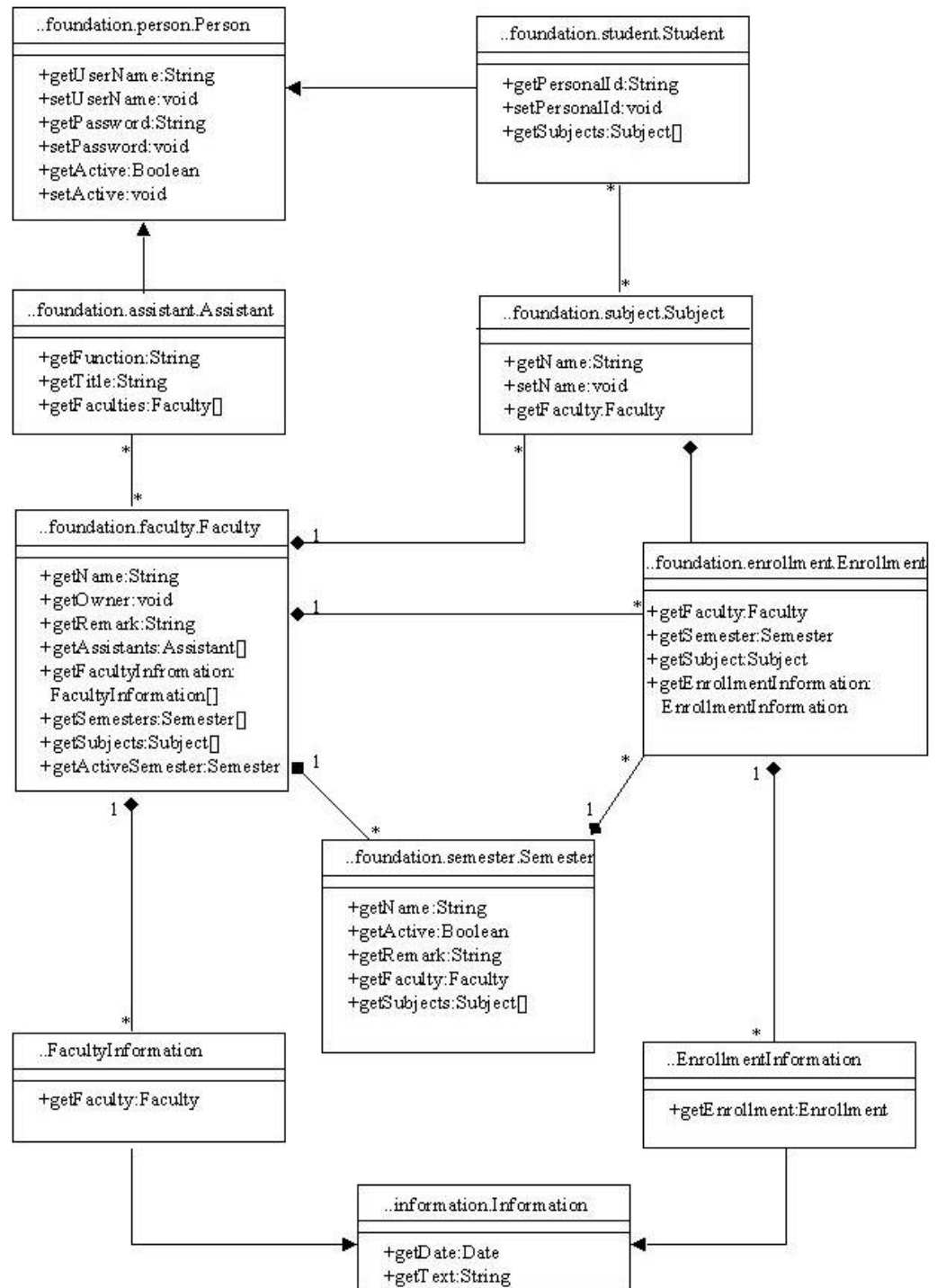


Figure 13

The class diagram in Figure 13 defines the relationships and dependencies between the foundation components of Business and Data Layers as follows:

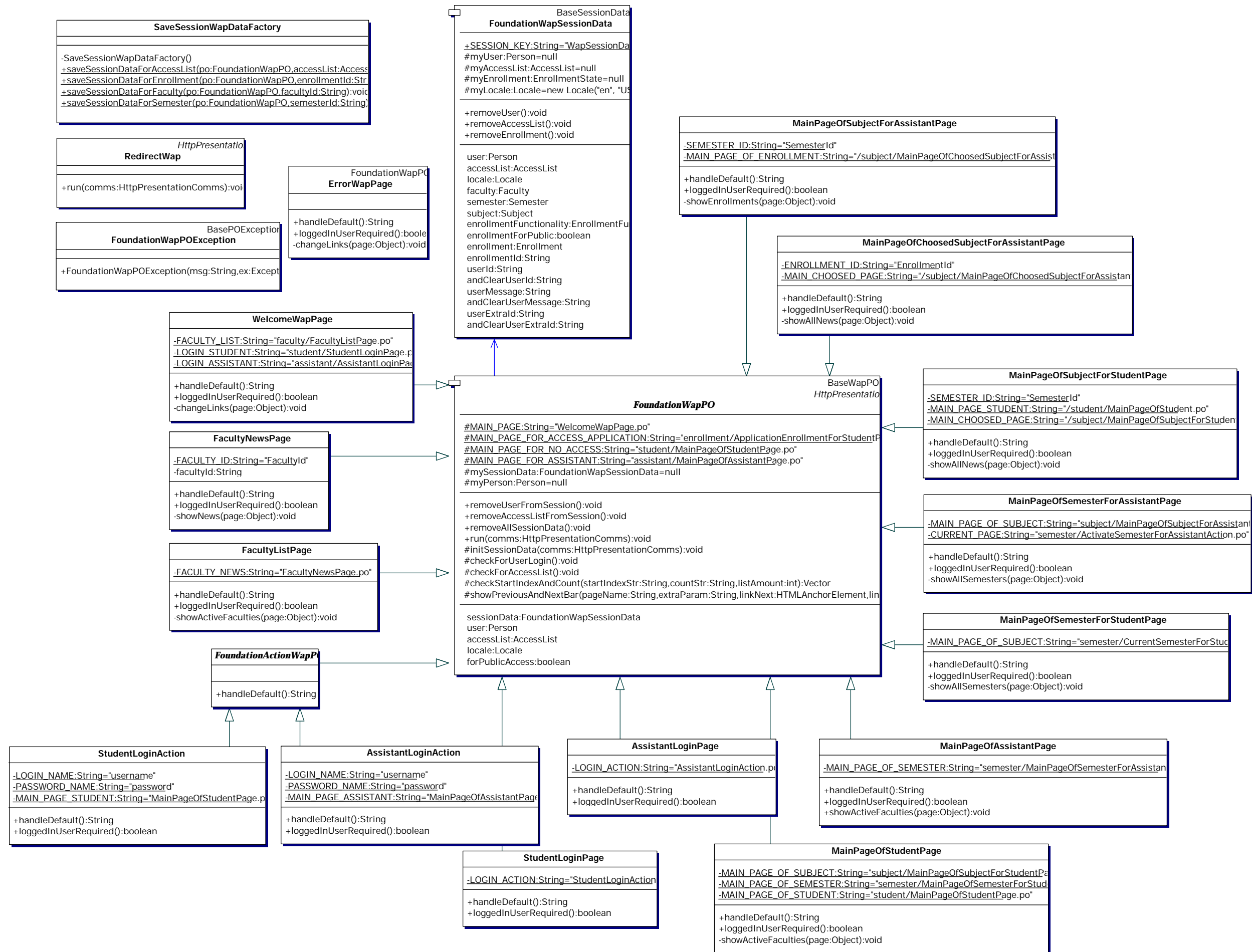
- Here class *Person* has a generalisation relationship with the more specific classes *Assistant* and *Student*.
- Class *Student* is independent from *faculty* because a student can register for subjects in different faculties. A student can have many subjects.
- Class *Assistant* can have access to more than one faculty.
- Class *Faculty*, *Semester*, and *Subject*. A faculty has components of semesters. One faculty can consist of many semesters and many subjects. Subjects can only be owned by one faculty.
- Class *Enrolment* plays an essential role to make sense of the relationship between a faculty, a subject, and a semester. This class is comparable with an order that shows a relationship between a customer, a product and a time point.

To be able to retrieve information from the Business Layer, Java servlets are made for showing the WML file on the WAP-Browser in the Presentation Layer. These Java servlets should invoke the methods of classes from the Business and Data Layers that are already made by OpenUSS. Servlet in Enterprise Application is primarily a co-ordinator. Hence, it should only take on lightweight responsibilities, which could include initiating some business logic. However, actual business logic, computations, interaction with entity objects etc, would all be the responsibility of the Business and Data Layers⁴⁶.

All the Java servlets of this implementation are shown in the next class diagram. The relationships between the classes next do not use the association notation considered in the previous class diagram, but a generalisation. Generalisation is shown as a line ending with a triangular arrowhead at a class, which is the more general type⁴⁷. For example, most of the classes are an extension of class *FoundationWapPO*, which is in this case the superclass of the subclasses *WelcomeWapPage*, *FacultyListPage*, *StudentLoginAction*, *LecturerLoginAction*, *StudentLoginPage*, *AssistantLoginPage*, etc. Generalisation allows the inheritance of the attributes and operations of a superclass by its subclasses. Given the stateless nature of the HTTP protocol, managing repeat interaction can be overcome by saving *cookies*. This is the responsibility of a servlet session, which is the class *FoundationWapSessionData*. These cookies store information that later can be passed back to the client repeatedly. The Web server provides the cookie to the browser. *SaveSessionWapDataFactory* is a factory object for creating some useful objects such as faculty, semesters or enrolments and saves these useful objects directly in the session. *SaveSessionWapDataFactory* uses *FoundationWapSessionData*. Classes *FoundationWapPOException* and *ErrorWapPage* are responsible for error message. The class *RedirectWap* is a ServletContext interface that can be used by servlets to store and retrieve information and share among the servlets. The following class diagram shows the implementation of this project:

⁴⁶ <http://www.therationaledge.com/content>

⁴⁷ Bennet, et al, 2001



4.4 Design and Implementation

Before starting with the design, I had to study the OpenUSS architecture thoroughly to be able to implement my part of the program. This includes a deep understanding of the concept of Java 2 Enterprise Edition and Enterprise JavaBeans, and of how the EJB products JOnAS and the servlets container Enhydra can support Wireless application in OpenUSS's system. This requires skills in programming Wireless Markup Language and Java for the servlets. Therefore, the abilities of Enhydra for wireless application were investigated.

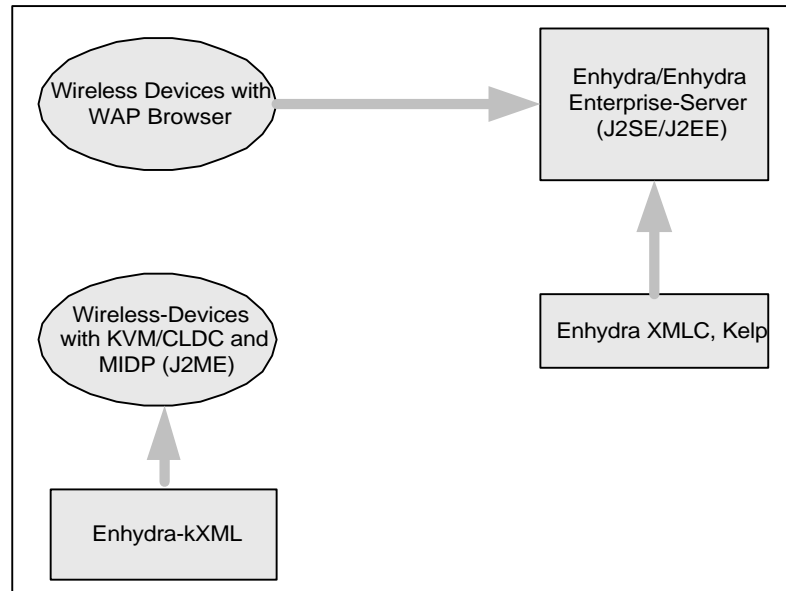


Figure 14

Figure 14 explains following types of wireless application provided by the Enhydra platform:

- Wireless application that holds a WAP-Browser. This application is comparable with the thin-client computer that is equipped with a Web-Browser. The application will be accomplished only on a server. The result – in form of HTML pages – will be given back to the client. Due to the small display of WAP devices a WML presentation format will be used for the client, instead of HTML presentation format. These WML-pages must be created on the server. In this case, Enhydra demonstrated its excellence.

Enhydra provides the XMLCompiler that process diverse XML-derivatives such as HTML and WML. This means that the creation of WML-pages is not complex.

- Local application for Wireless devices. This application is comparable with the fat-client concept of a client/server technology. All the applications run locally and primary offline on the wireless devices. For such application the wireless devices must provide kVirtual Machine/Connected Limited Devices Configuration (KVM/CLDC) and Mobile Information Device Profile (Java 2 Micro Edition)⁴⁸.

⁴⁸ CLDC is a specification that outlines the most basic set of libraries and Java Virtual Machine features that must be present in each implementation of a J2ME. KVM is a Java virtual machine, which is specifically designed for resource-constraint consumer devices. It is included in the CLDC reference implementation. < <http://java.sun.com/products/cldc/faqs.html#2>>

To enable the implementation, OpenUSS system was installed. This includes JOnAS that acts as a Enterprise JavaBeans container and Enhydra as a servlet container. Those are Java environment programs that mean that any operating system can be chosen as long as it supports Java Virtual Machine. This means that Java Development Kit (JDK) must be installed first. OpenUSS distribution already includes and installs all the components, which are needed, automatically. This means: JOnAS, Enhydra and HypersonicSQL are included within OpenUSS distribution, no need to download them separately.

To provide the Data Layer, HypersonicSQL is installed automatically when downloading OpenUSS. It is used as a default database for testing purposes. OpenUSS allows using other databases, which have to be customised with OpenUSS by the developer manually. In this case, I choose to use the convenience that OpenUSS offers.

For running business objects in the Business Objects Layer, JOnAS version 2.4.4 is installed and for the Presentation Layer, Enhydra 3.1 is installed.

Since OpenUSS uses JOnAS, of OpenUSS breaks the application up into three categories of objects: presentation objects, business objects, and data objects.

Presentation objects handle how the application data is presented to the clients (Web or WAP browsers). Any and all HTML/WML is kept in the presentation objects. Data objects get and set the data that application manipulates, from a file or a database. All database code, or file reading code, is kept in the data objects. Business objects handle all the "business logic". That is all the policy decisions, algorithms and data manipulation. Essentially everything left over after the entire HTML and WML to the presentation objects, and all the database/file access code to the data objects are quarantined⁴⁹.

Figure 15 shows the OpenUSS multi-tier architecture using JOnAS and Enhydra⁵⁰.

⁴⁹ <http://enhydra.enhydra.org/software/documentation/EnhydraApp.html>

⁵⁰ <http://openuss.sourceforge.net/openuss/developer/groups/architecture/architecture.html>

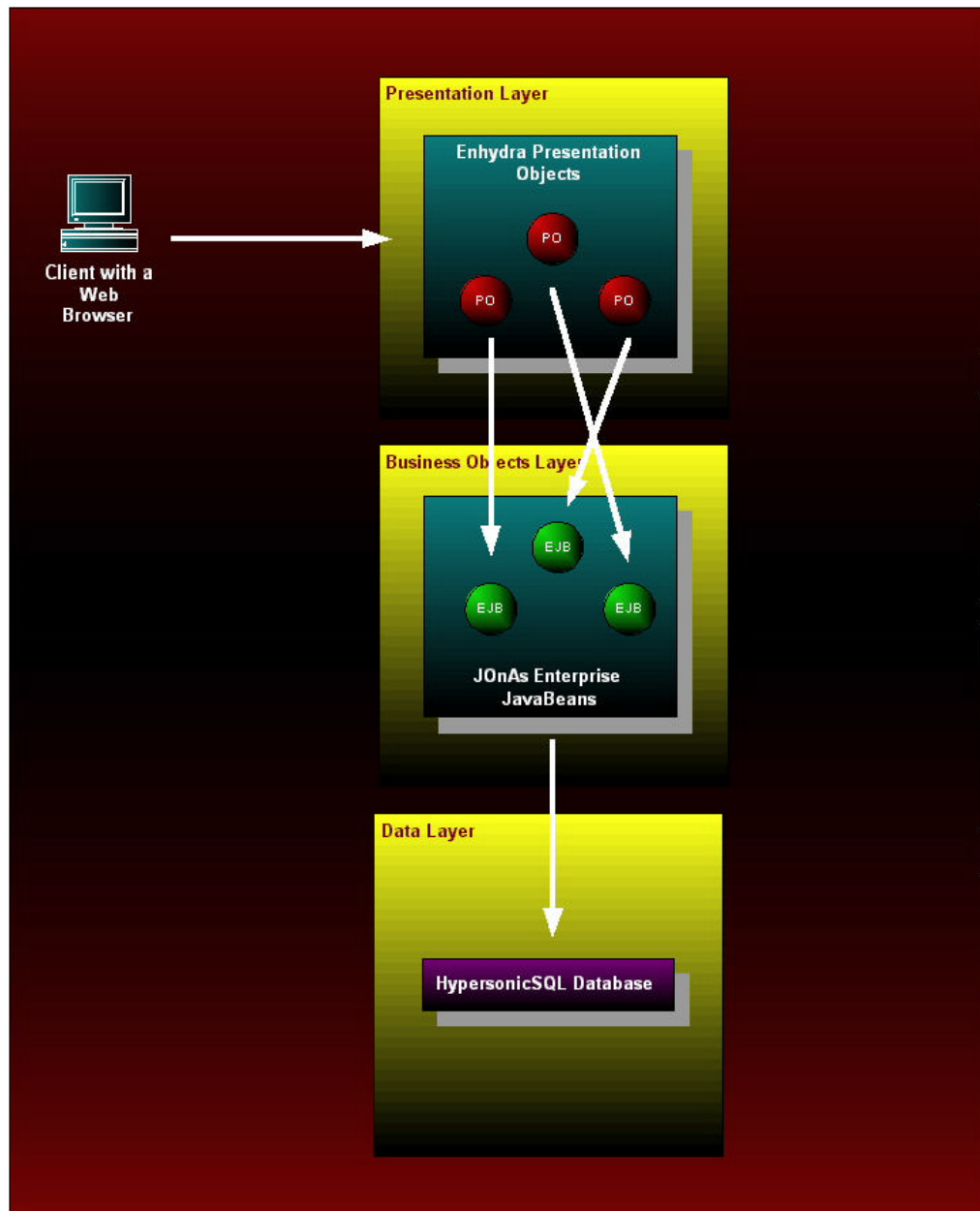


Figure 15

As mentioned in the previous Chapter, Enhydra has features like Servlets and XMLC. This means Enhydra also support a fully functional WML Data Type Definition. Thus, WAP application with complete dynamic functionality in WML and Java can be written.

Principally, there are three steps to engender dynamic WAP application, they can be describe as follow:

1. The definition of the dynamic WML-Components: Here XMLC provides the ID property in the tags. In the example below (*FacultyList.wml*) we have five *ids*: `FacultyList`, `FacultyListPElement`, `FacultyListAElement`, `FacultyListMorePElement`, and `FacultyListMoreAElement`. The first id is not interesting for our purpose since it is a basic unit of WML. It is the card that specifies a single interaction between the user and the user agent. Multiple cards are grouped together in decks.

A deck is the topmost element of a WML document. When the user agent receives a deck (by downloading the complete deck), it activates only the first card in the deck. Our example only shows a sample WML document with a single card. This `<card>` element contains a `title` attribute, which the browser will display for the card. The `<p>` tag delimits a paragraph and must be used to wrap any displayed text. It also allows to control the layout of text with its `align` attribute. The `<a>` tag is used to link a text and is specified in the `href` attribute. In this example, any Faculty has a link to its news. The *ids* with the `<a>` tag will be evaluated all the time and will be substitute through recent datas such as Computer Science, Electronic Engineering for *FacultyList*. A more button is added to show the lists that more than five. Following is the template for FacultyList:

```
<?xml version="1.0"?>
<!DOCTYPE wml
  PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
  "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>

  <template>
    <do type="options" label="Back">
      <prev />
    </do>
  </template>

  <card id="FacultyList" title="Faculties">
    <p id="FacultyListPElement" align='left'>
      <a id="FacultyListAEElement" href='FacultyNews.wml'>Institute
for
  Economic and Social Science</a>
    </p>
    <p id="FacultyListMorePElement" align='right'>
      <b><a id="FacultyListMoreAEElement" href='FacultyList.wml'>
more..</a></b>
    </p>
  </card>

</wml>
```

WML is based on XML. Therefore, a deck has to be a valid XML document, which implies that a WML document (as the one shown above) should start with the standard XML header and the reference to the WML DTD (Data Type Definition).

2. Enhydra then uses its supplied XMLC program to parse and compile the pages into properly formatted XMLC classes. As a result, a Java-DOM-Class *FacultyListWML.java* is created. As mentioned above, DOM allows access to XML files so that the XML Compiler will also allow for presentation objects to serve dynamic content XML. It is used to provide access methods to read and modify the content. In order for the engineer to alter the dynamic content of a page, id and class attributes are added as dynamic tags. The attributes, content, and nested tags can then be replaced or removed by the Java program.

Figure 16 below shows how the Enhydra XMLC works:

XMLC/HTML

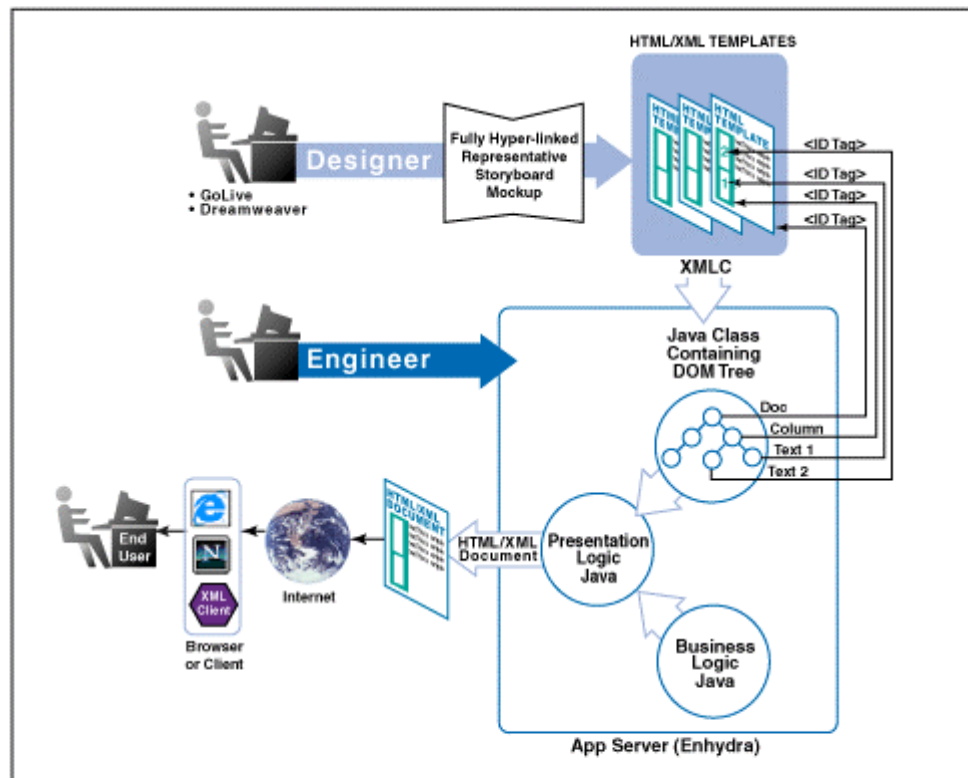


Figure 16

- Corresponding Java classes (servlets) must be created for each WML page that will be shown on the WAP devices. This will load up the XMLC classes, does whatever dynamic stuff you tell it to do, and with a call to the function to serve up the page. Enhydra presents the page as **[page name].po** in the browser, which stands for Presentation Object. To the user, it looks just like standard WML. The servlet of the “Faculty List” (*FacultyListPage.java*) can be seen as follow:

```
public class FacultyListPage extends FoundationWapPO {

    // Constants
    private static String FACULTY_LIST_PAGE = "FacultyListPage.po";
    private static String FACULTY_NEWS = "FacultyNewsPage.po";
```

By implementing the method below, no more than five lists of faculties will be shown. If there are more than five faculties, a „more...“ button will appear to show the rest of the lists:

```
private static String START_INDEX = "startIndex";
private static String COUNT = "Count";
private static int LIST_AMOUNT = 5;
```

The code below uses the convenience methods generated by XMLC, such as `getElementFacultyList`, to access the table cell in the DOM tree.

```
// Work through the faculty table and change the content in this
// table
// Create the WML template references and always remove the
templates
WMLCardElement facultyListCard =
    welcomePage.getElementFacultyList();
WMLPElement facultyListP =
    welcomePage.getElementFacultyListPElement();
WMLAElement facultyListA =
    welcomePage.getElementFacultyListAElement();
// Remove...
facultyListP.removeChild(facultyListA);
```

Another important issue is showing all faculties that exist. All the lists of faculties is invoke by implementing following methods:

```
// Show all faculties already exist.
private void showActiveFaculties(Object page) throws
HttpPresentationException {
FacultyListWML welcomePage = (FacultyListWML)page;
```

The `Vector` acts similar like an array, with the `findActiveFaculties()` method all active faculties from business layer of EJB will be retrieve:

```
// Get collection of faculties
FacultyHelperObject facultyHelper = home.create();
Vector facultyList =
facultyHelper.findActiveFaculties(startIndex, count);
Iterator facultyIterator = facultyList.iterator();
```

Using the while loop, the wml template will be inserted with the real data, which is taken by the EJB server. Working with wml template is the same as with working as DOM XML parser:

```
// And loop through collection
while (facultyIterator.hasNext()) {
// Get the current faculty
currentFaculty = (Faculty)facultyIterator.next();
// Deep copy of AElement, not shallow
WMLAElement copy = (WMLAElement)facultyListA.cloneNode(true);
// Set the link for the news
copy.setHref(FACULTY_NEWS + "?FacultyId=" +
currentFaculty.getId());
copy.setId("link" + currentFaculty.getId());
copy.getFirstChild().setNodeValue(currentFaculty.getName());
// Now add this to our list of links
facultyListP.appendChild(copy);
// Now create a <br/> tag and append it after the link
facultyListP.appendChild(welcomePage.createElement("br"));
}
```

Subsequently, the JOnAS server and the Enhydra server can be started. The access can be made in the WAP-browser simulator to `http://localhost:9005`. Note that the first time an Enhydra application is called, it takes a while for it to come up (a minute or so). After that it is fast.

The following screenshots shows (Figure 17) that the same information on the HTML-Web browser with the WAP browser:

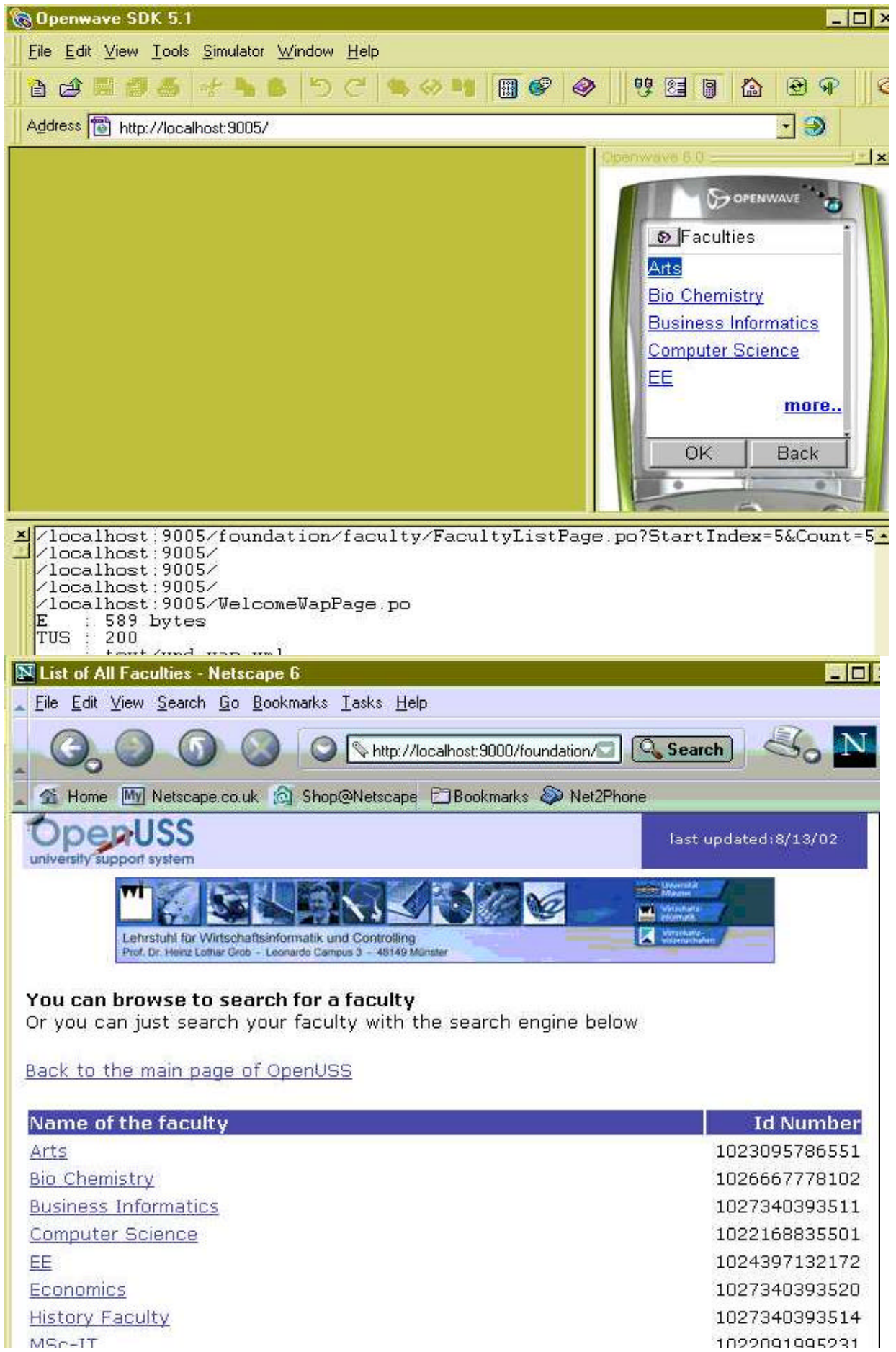



Figure 17

Another issue that is important is to present an image on WAP. Unfortunately, the only graphic file format that is currently supported by the WAP Forum is the *wireless bitmap* (WBMP) format. It is simply a one-bitmapped image.

A converter is needed to convert a BMP image to a WBMP image. An online converter was used where can be found at <http://www.teraflops.com/wbmp/>. The converting process is fairly straightforward. The image file can be inserted in within the <p> tag. In the *WelcomeWap.wml* for example:

```
...
<card id="WelcomeList" title="Welcome">
    <p align='center'>
         </p>
    <p>
...

```

The original OpenUSS logo  with the one shown the simulator screenshot using Siemens S45 can be compared in Figure 18:

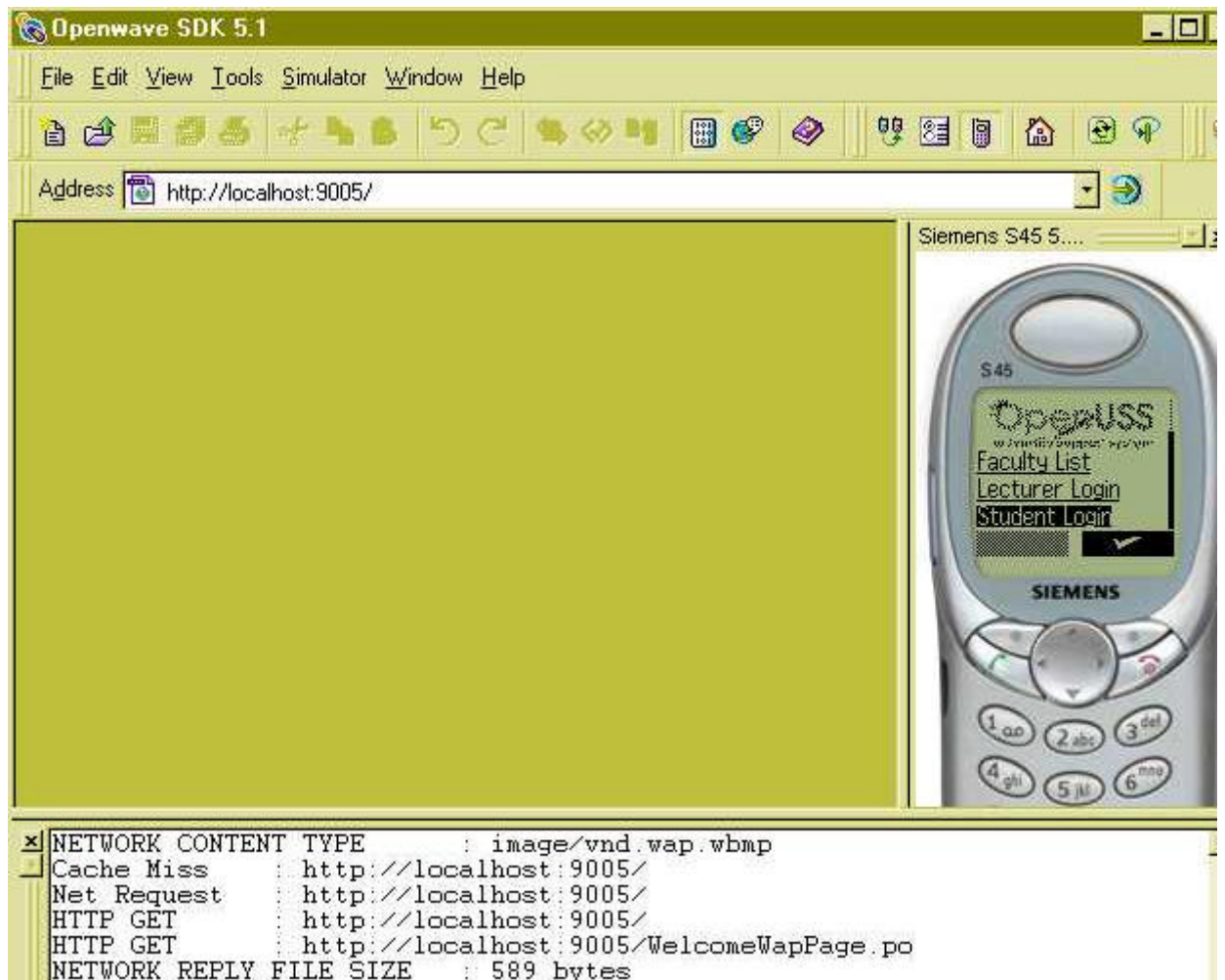


Figure 18

4.5 Testing and Evaluation

To prevent bugs and system failures, in-depth tests and thorough evaluations have to be carried out. This involves the entire software development PROCESS - monitoring and improving the process, making sure that the requirement standards in the requirement analysis meet their need, and ensuring that problems are found and dealt with. In anticipation of the limited time and resources given, testing was carried out during the entire implementation phase. The following problems and appropriate solutions were found:

- **Problem:** The first simulator version from Phone.com, UP.SDK 4.0 was used. This caused an abrupt system crash of my computer for no apparent reason. This simulator provided different types of phones such as Motorola and Alcatel handsets. When the device configuration was changed, e.g. from Motorola to Alcatel, the application stopped running and sometimes crashed the whole system. This was very annoying because I had to reboot the whole system.

Solution: I downloaded a recent version of the Openwave development kits (Openwave version 5.1). This included a phone simulator (built on the same browser code that is embedded into commercially shipping phones such as Siemens S45), WAP gateway simulation (that uses the same code as it found at the operator sites), and debugging tools. The 55,5 MB development kits can be installed without problems. It integrated itself automatically when it was installed in the same classpath as Java, JOnAS and Enhydra. In this case, I installed all of them on my hard drive.

Since WAP gateway simulation is provided, a WAP gateway is not needed in this test. All Openwave SDKs have two modes of operation⁵¹:

1. HTTP Direct, which simulates the WAP gateway; and
 2. A gateway or proxy mode that connects the SDK to the content site via a real WAP gateway.
- **Problem:** Because the WAP Browser from Openwave 5.1 accepts only a maximum of 1500 to 2000 bytes per deck, it can last over 10 seconds. Unlike Web users that can browse into other windows while waiting for a page that takes a long time, wireless devices prohibit users from performing other tasks on the device while waiting for content to load. The perceived network latencies can vary.
Solution: To keep the application moving smoothly, each deck should be kept as small as possible. The recommended guideline is 500 bytes or less per deck (encoded WML)⁵². Testing on the Openwave 5.1 simulator enable me to use the Phone Siemens S45. In the real world, this device supports GPRS, which means that the WAP access is truly faster. Furthermore Siemens S45 has 48 kBytes of memory.
 - **Problem:** Starting the Openwave 5.1 program. Because I use AOL as Internet Service Provider, the connections in the Internet Properties in the control panel are set automatically to “Always dial my default connection”. Every time the simulator is started, AOL is launched automatically. This occurs even when connecting to the localhost <http://www.localhost:9005>.

⁵¹ http://www.openwave.com/products/developer_products/sdk/sdk_faq.html

⁵² Forte, et al, 2000.

Solution: Every time before I start the simulator, I have to set up the connections in the Internet Properties in control panel as “never dial a connection”.

Bugs: virus-checker programs can affect SDK performance. Fortunately, the anti-virus program in my computer does not affect the performance of this system.

- **Problem:** Using <go> task to navigate to a card that is already on the history stack. This task returned to the subscriber homepage such as <http://openwave.com>.

Solution: Use the <prev> task to navigate the backward direction. Most of the WML files in this system use following command:

```
<template>
  <do type="options" label="Back">
<prev />
  </do>
</template>
```

Furthermore, there is an option to configure the homepage of the device (<http://localhost:9005>) that allows the phone to directly apply to the application servers by using the HTTP direct.

- This problem and solution is recommended from the Openwave.com: “When content server updates the value of a cookie, SDK still transmits the previous value in the HTTP cookie header. In direct mode, the workaround is to reload the device after the value of the cookie has been updated, which will force the SDK to use the updated value of the cookie. Another workaround is to cause the content server to set the new value of the cookie twice. Since the SDK always uses the previous value of a cookie after it has been updated, updating the value of the cookie twice (with the same values) will cause the SDK to use the correct value”⁴⁸.

The link to the “Faculty List” works well. A **more...**-button provided when the list is more than five items. The selection of users and the login function work well. Access to the information demanded can be done easily. **Back**-button is provided on each page, even on the Login Page, depending on the devices, there is an edit button. To get back more quickly from the Login Page (if user avoids editing), a back button is also provided.

Testing whether both presentations (HTML and WAP) work simultaneously, both localhosts (<http://localhost:9000> for HTML and <http://localhost:9005>) for WML have to be run together. The information changed on HTML such as adding or deleting news, can be seen on the WAP-Simulator straight away with one condition: a new session has to be started.

Mr. Dewanto, the developer of OpenUSS, put my WAP implementation on the server of OpenUSS for testing purposes with the URL: <http://pcwi505.uni-muenster.de:9021/>. One can test it with a simulator from the Internet that support cookies or with a WAP phone without the *http://*. As a result, it works really well. The GUI depends on the WAP-devices. Everybody who tested the system was amazed. It looks simple and efficient.

5 Future Development

From the discussion in system testing and evaluation, it has been shown that the system has to some extent fulfilled the user and system requirements and has therefore achieved our objectives. However, future improvements can be made to enhance the usability and functionality of the system:

- Implement a discussion page so that the user can see what has been discussed in the forum. This has not been made because of:
 - I. the extension foundation feature of OpenUSS system that require more than writing a servlet i.e. change the business logic.
 - II. the limited time given for this project.
- Although it is unlikely that one student will take more than ten subjects in one semester, it could be useful to implement a ‘more...’ button in the subject list both for student and assistant. This is due to the OpenUSS system that needs to extend its Business Logic in the Business Layer. As state previously, this project only deals with the Presentation Layer so that the Business Layer is fall out of the theme.
- Implement a direct button to go back to the main menu. This could be unnecessary, because by doing this the user will log off i.e. the main page consists of ‘Faculty List’, ‘Lecturer Login’, and ‘Student Login’. It could be necessary if the user wants enter the ‘Faculty List’ directly from the main menu.

6 Conclusion

Although there are critical issues concerning WAP, special attention must be paid to the limitation of the devices. There has been a lot of effort put into the development of mobile devices. When I went to the world's leading trade show for IT, telecommunications & networks CeBIT in Hannover (Germany) this year, it showed lots of development in this area. For example the astonishing Pocket PC with Personal Digital Assistance functions, mobile Internet and GPRS, all in one. Or a Motorola UMTS handset prototype with colour display. With GPRS and UMTS technology, speed, stability, and display quality of WAP devices can be expected to improve. Without doubt, the improved technology of the WAP bearer services (GPRS, UMTS) will make WAP far more attractive.

This system makes full use of Open Source software. Together, JOnAs, Enhydra, and Openwave 5.1 represent a complete recipe for the success of OpenUSS-mobile development.

Finally, I believe that “Mobile Software Development for an Open Source E-Learning Platform” is an open, strategic, long-term software solution, and therefore will easily achieve widespread acceptance in the e-learning environment.

Bibliography

- BENNET, S., Skelton, J., Lunn, K. *Schaum's Outlines of UML*, 2001.
- Bensberg, F., Dewanto, L.: *Open Source Java-Enterprise: Plattformen, Mediatoren und Communities*. Java Magazin (4.2001), P. 74-80.
- Bensberg, F., Dewanto, L.: *Seeotter trifft Pinguin! Ein Überblick über die Enhydra-Plattform*. Linux Enterprise (7.2001), P. 46-50. . (in German – Translation of title in English “An Overview about Enhydra”)
- Bensberg, F., Dewanto, L.: *Und wo bleiben die Gewinne? – Open-Source-Software: Wirtschaftliche Aspekte unter der Lupe*. Java Magazin (4.2001), P 82-85 (in German – Translation of title in English “ And where is the profit left?”)
- BIGELOW, K. and Beaulieu, M. Lutris Ltd. *Wap vs. i-mode vs J2ME Programming Paradigms and Limitations* [online]. Available from World Wide Web: http://www.sdc.sun.com/briefings/pdf_files/KeithBuildingMultichannelApplicationswithXMLandJ2ME.pdf
- Communications International. *Riding the third wave – Japan has already found out how to make 3G work. Can Europe?* February 2002
- DEITEL, H.M., Deitel, P.J.: *JAVA HOW TO PROGRAM*, Fourth Edition, 2002.
- DEWANTO, Lofi. *Developer's Manual OpenUSS OpenUSS-university support system*, 2000 [online]. Available from World Wide Web: <http://sourceforge.net/projects/openuss/>
- Glahn, Kay: *WAP ist ein Prozess*. Ein Gespräch mit Scott Goldman, CEO WAP Forum Ltd. Java Magazin (11.2001), P. 12-16 (in German – Translation of title in English “WAP is a process”)
- Glahn, Kay: *UMTS, WAP und Co: Technologien für das mobile Internet*. Java Magazin (3, 2001), P 55-59 (in German – Translation of title in English: “UMTS, WAP and Co: Technologies for mobile Internet”)
- Glahn, Kay: *WAP ist langweilig*. Interview mit Curtis Sasaki, Director of Technology Advocacy bei Sun. Java Magazin (9.2001), P. 18-20 (in German – Translation of title in English: “WAP is boring”)
- FIELDEN, T, Orubeondo, A., *J2ME and WAP: Together forever?* [online]. World Wide Web <http://www.javaworld.com>
- FORTA, B., Lauver, K., Fonte, P., Juncker, R.M., Mandel, R., Bromby, D., *WAPDevelopment with WML and WMLScript*. The Authoritative Solution, 2000.
- Haiges, Sven: *GeWAPnet*. Java-Architecture eines modernen WAP-Services. Java Magazin (9.2001), P 69-72
- ROMAN, Ed, Ambler, S.W, Jewell, T., *Mastering Enterprise JavaBeans*, Second Edition, 2002.
- ROMAN, Ed, *Mastering Enterprise JavaBeans and the Java 2 Platform Enterprise Edition*, 1999. Book on CD-ROM

- SAVARESE, Daniel F.: *The Great Migration*. Java Pro, September 2001. P.20-21, P.88-89
- SUN Microsystems, Inc. *Simplified Guide to the Java 2 Platform, Enterprise Edition* [online]. Available from World Wide Web <<http://java.sun.com>>
- PEARCE, James: *WAP for web developers*. (Part 2). February 2000 [online]. Available from World Wide Web <<http://www.anywhereyougo.com/Content.po?name=wap/Fivesteps>>
- POUNTAIN, D., Montgomery, J.: *Web Components*, August 1997, P.56-68.
- PRESSMAN, Roger S., adapted by Darrel Ince, *Software Engineering – A Practitioner’s Approach – European Adaptation*, Fifth Edition, 2000.
- WAP-Forum. Available from World Wide Web: <<http://www.wapforum.org>>:
 - *Releases WAP 2.0 Specification For Public Review* [online], 1st August 2001
 - *WAP 2.0 Technical White Paper* [online].
 - GOLDMAN, Scott. CEO WAP Forum. *Status of WAP* [online]. Sydney, 15th May 2001
 - GOLDMAN, Scott. CEO WAP Forum. *The Future of WAP* [online] London May 2001
- WITT, Martin, *GPRS Start in die mobile Zukunft (GPRS-Start in the mobile future)*, 2000. (in German – Translation of title in English “GPRS-Start in the mobile future”)
- WOOD, David. *Java Application Servers*, 1997 [online]. Available from World Wide Web: <<http://www.pisoftware.com/publications/JavaColumn/appservers.9902.html>>
- WOFFENDEN, Claire. *iMode to merge with Wap, say pundits*, 2000[online] . Available on Web <<http://www.computing.co.uk/News/1106079>>
- *WML versus cHTML* [online]. Available from World Wide Web <<http://www.netlight.se/imodevswap.html>>
- <http://www.opensource.org>
- <http://www.openuss.sourceforge.net>
- <http://www.objectweb.org>
- <http://www.perens.com/Articles/OSD.html>
- <http://www.enhydra.org>
- <http://www.gnu.org>
- <http://java.sun.com>
- <http://www.trolltech.com>
- <http://www.kde.org>
- <http://www.umtsforum.org>

- <http://www.openwave.com>
- <http://www.eplus-imode.de/1/de/html/pub/presse/index.html>
- <http://www.openuss.org>
- <http://www.handytel.com>
- <http://www.javamobiles.com>
- <http://netlight.se>
- <http://www.cellular-news.com>
- <http://www.mech.soton.ac.uk>
- <http://www.cs.waikato.ac.nz>
- <http://news.bbc.co.uk>
- <http://java.unidata.ucar.edu>
- <http://www.mobilinkgsm.com/wap>
- <http://www.webtechniques.com>
- <http://www.provu.co.uk>