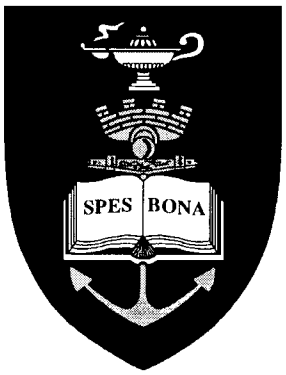
A BLUETOOTH EDUCATIONAL CONTENT DISTRIBUTION SYSTEM MODELLED ON A SERVICE-ORIENTED ARCHITECTURE

(Kamulegeya Grace Bugembe)



This thesis is submitted in partial fulfillment of the academic requirements

for the degree of

Master of Science in Computer Science

in the Faculty of Science

University of Cape Town

November, 2008

CERTIFICATION

As the candidate's supervisor, I have approved this dissertation for submission.

A/Professor Gary Marsden

Signature: Signed by candidate

Date: 22/10/09

DECLARATION

I declare that this thesis is my own work. Where collaboration with other people has taken place, or material generated by other researchers is included, the parties and/or materials are indicated in the acknowledgements or are explicitly stated with references as appropriate.

This work is being submitted for the Master of Science in Computer Science at the University of Cape Town. It has not been submitted to any other university for any other degree or examination.

Signed by candidate

29/09/2008

Kamulegeya Grace Bugembe **Date**

Contents

CONTENTS

LIST OF FIGURES

ABSTRACT

In this research, we design and prototype an educational content distribution system modeled on a Service-Oriented Architecture (SOA) paradigm and implemented using Web services, XML and Bluetooth technology. In the prototype, we use an Open Source Learning Management System (LMS) Sakai implemented in Java and branded Vula for the University of Cape Town (UCT). Web services and its specification of SOAP, XML and Bluetooth technology are used to integrate the disparate technologies that form the service architecture. The disparate technologies include among others Bluetooth enabled mobile phones and PDAs, services (modules) which may be running on different operating systems, and deployed over Local Area Networks (LANs) or Internet. The service is meant to leverage the existing infrastructure to provide a new, cheap channel for education content distribution to mobile devices in learning institutions especially Universities in the developing world and Africa in particular. We design, implement and evaluate the prototype for performance and scalability. During the designing and implementation of the architecture, we incorporate SOA principles of service/module re-use, service composition, loose-coupling, standard data exchange within the system or services, and extensibility of the services among others. The aim of the service is to distribute education content uploaded in Learning Management Systems (LMSs) to Bluetooth enabled mobile devices that are increasingly held by students in developing world Universities. The service is intended to supplement existing Web-based and lecture room content distribution channels by opening up the mobile device space. For the prototype, we focus on repackaging structured text content and distributing it to Bluetooth enabled phones and PDAs using Bluetooth technology. We evaluate our prototype for performance using experimental studies.

ACKNOWLEDGEMENTS

This research would not have been possible for me had it not been for the profession guidance, critical advice, and financial assistance provided by my supervisor, A/Prof. Gary Marsden. Thanks go to Dr. Fred Kiggundu and family for the love and constant encouragement, a loving atmosphere and immense financial assistance they have always provided to me throughout my education. Special thanks go to Dr. Edith Kiggundu for the emotional encouragement when many things seemed to be still. Thanks are also due Mrs. Prossy Nalule Kasozi for being the mother throughout my formative years and for providing through my early education. Thanks to my sister Joan Nattabi for always keeping me warm during the cold winters. Special thanks go to Dr. Michelle Kuttel for critical comments made during my research proposal presentation. Thanks go to my friends at the CVC laboratory at the University of Cape Town, Ming Ki Chong, Samuel Olalekan, Calvin Pedzai, Shikoh Gitau and Andrew Maunder, for the day and night laboratory time we shared. Thanks to David Horwitz for the technical advice and assistance about Sakai and Vula in particular, in the early stages of my system implementation. I dedicate this dissertation to my beloved late mother Justine Bugembe, my late Father Eliphaz Bugembe, and my late Brother Thomas Mukasa Kagombero.

1. INTRODUCTION

Higher learning institutions, especially universities, in the developing world and Africa, in particular, are facing ever increasing student enrolments as highlighted by Sawyerr (2004). The large numbers of students are straining the already existing physical infrastructure, technology resources and staff capacity. This has led universities to devise more efficient ways of utilizing the existing resources without having to invest heavily in new ones. In a bid to involve the students in learning activities, many universities are using Learning Management Systems (LMSs) (sometimes called Course Management Systems (CMSs)) to extend learning beyond the limitations of physical boundaries and staff capacity. These LMSs offer many functions from day to day learning activities to administration (which students' access using PCs and laptops).

There is, however, a need to make these LMSs accessible from mobile handsets as these devices, in the developing world at least, are possessed by more students than PCs and laptops. The popular use of mobile devices by students in developing world universities reflects their high prevalence as a technology among the general population. Statistics show that there is a high penetration rate of mobile devices in the developing world compared to PCs and landline telephones. According to the International Telecommunication Union (ITU) statistics, by the end of 2007, 45 out of 100 inhabitants in developing world had a mobile phone. This constitutes more than 1 out of 4 Africans and 1 in 3 Asians. Fixed land lines in comparison had a penetration of 3 per 100 inhabitants. Internet use remains very low too, in Africa with 5% of the African population (International Telecommunication Union, 2008).

STRAINED RESOURCES

Physical infrastructure like classrooms, laboratories and libraries can hardly accommodate the students in a single course seating, often leading to division of classes and extension of lecture time. For example, some universities are offering evening classes in addition to day classes (Musisi & Muwanga, 2003). In some universities extra enrolled students are taught in virtual (online) classrooms away from university campuses.

Technology resources are also being strained by the student numbers. This has led to time rationing of the few resources like computers, printers and, in some universities, students are also

being capped on the amount of Internet bandwidth they can use per month. They may also be restricted on the type of content they can download or upload from the Web.

In these universities too, learning material like books, journals and other print media are shared amongst students and are sometimes not enough to aid students in their day-to-day learning activities. To alleviate the problem of sharing printed media, universities are creating electronic versions of books, journals and hosting them in online libraries and/or LMSs. The electronic copies can then be more easily shared by students than the hard copy versions. But these electronic documents still need efficient technology for distribution to the students. Thus, their distribution to students is still limited by available technology resources like Internet bandwidth and computers.

ADOPTING LEARNING MANAGEMENT SYSTEMS

To overcome some of the problems created by limited resources, LMSs are being used and are becoming popular amongst many universities. They provide a number of benefits that overcome many of the physical infrastructure and staff capacity constraints. Firstly, most LMSs are accessible using Web-based interfaces and thus can be accessed within and outside university physical boundaries thus, breaking down the traditional space-constrained settings of classes, libraries and laboratories. Students do not have physically to be present in class to receive lecture notes, upload assignments, and write tests, view results, get course announcements and timetables, share documents and do discussions. These learning activities are offered by most LMSs regardless of location (since most are online based).

Apart from the 'out of class' enabling of study activities, most LMSs offer other benefits that include, student self assessments, self course enrolments, content authoring, activity scheduling, batch student registrations, detailed reporting, student course feedback, to mention but a few. All these benefits have thus contributed to their popular adoption in many learning environments in developing world institutions.

LIMITATION OF CURRENT LMSS

However, most of the benefits of LMSs are realized through the fact that they are accessible online using Web-based interfaces. The use of Web-based interfaces implies that PCs/laptops

with Internet and/or network connections have to be used. If the LMSs are to be accessed via mobile handsets, then the desktop Web-based interfaces have to be redesigned to fit the limitations of these devices. This redesign may lead to a loss of some of the LMSs capabilities and hence benefits. The use of Internet and /or network connections to link the PCs and laptops to the LMSs means that network resources like bandwidth are still a limitation. The cost of hardware and bandwidth will therefore still limit the use of the LMSs in developing world universities since many students cannot afford personal PCs/laptops and those who can afford then are limited by the available network bandwidth.

Single-user interfaces to access Web-based LMSs, mean that the available PCs and laptops in universities have to be time rationed so that as many students as possible can use them. To alleviate problems associated with these interfaces, alternatives like shared displays have been devised. These allow more than one student to access the same screen to download or upload content. Shared displays provide benefits that include shared access to Web-based content, collaborative learning (since they are large enough to hold multiple working areas), high computational power, and 'walk up' use through, say, camera and Bluetooth mobile phones, and touch (Paek, et al., 2004; Russell, Drews, & Sue, 2002). The shared displays, however, are still expensive (require expensive display hardware) to be deployed in most developing world universities, and most require a user to have a mobile handset enabled with a still camera or video camera and Bluetooth or infrared technology. Handsets with a cocktail of these features are still expensive compared to those without and may make them less affordable by students. Shared displays are also required to be installed in areas where a user has a clear view of the screen without being obstructed so that they can effectively interact with them.

CURRENT USE OF MOBILE DEVICES WITH LMSS

Some LMSs can also be accessed via mobile handsets so that many more students can access them. The most prevalent form of this mobile access to learning material and studying purposes uses the Short Message Service (SMS) and may not necessarily be accessing content from LMSS (Cheung, 2004; Horstmanshof, 2004). These mobile handsets use paid subscriber networks and the student has to incur the additional cost of the SMS to access the LMSs through these commercial networks. Although the Web-based LMSs still require paid Internet/network access (by the institution), the student does not incur any additional costs to access them. Extending the

LMSs to mobile phones using SMS introduces an extra cost to the student and thus makes this alternative expensive. This alternative, though, allows more students to potentially have access to the LMSs (if they can afford the SMSs) since almost all mobile handsets can send and receive SMSs and do not have to possess extra features.

In other LMSs that use mobiles, clients are developed and deployed on mobile handsets that then have access to the LMSs. Systems that use this alternative are limited by the fact that not all mobile devices support the deployed clients and that different clients have to be developed to fulfill mobile device proliferation; an almost impossible task. This diversity limits accessibility to the LMSs to students who have those handsets that support the developed clients.

Most of the available solutions to increasing accessibility to LMSs through mobile handsets have their limitations, most notably added cost of connections and probable discrimination of some mobile handsets that do not support developed client interfaces. Some of those that require no mobile clients, but use shared displays, require more advanced handsets in addition to the expensive shared display hardware and its deployment requirements. These limitations still curtail the use of LMSs beyond the Web-based interfaces which require PCs, laptops and network connections, even though these resources are still expensive in developing world universities.

OUR OFFERING

Take a use scenario where a student, Jim, is registered for first year Psychology and Computer Science courses and has also registered the Bluetooth ID of his mobile phone for those courses with a content distribution node, to receive content from the university LMS to his phone. Getting back from a field trip, Jim has missed his lectures for the week and does not know the reading references and the results from the group experiments the class carried out. However, as he walks through the library, a Bluetooth content distribution node detects his mobile phone ID and queries the LMS for the results and reading references posted during the week, which it pushes to his phone.

Given the possession of personal mobile devices especially phones by students and the existence of Learning Management Systems, there is already an existing technology infrastructure in these universities, though it is disparate and may be currently underutilized. The above use-scenario

exemplifies a case in which mobile phones and LMSs can be integrated for use in tertiary institutions. These mobile handsets can be exploited as computing infrastructure onto which the existing educational material can be distributed. This may increase the penetration of educational material, thus increasing the currently existing Web-based content distribution through LMSs. This content can be in text and multimedia formats.

With Bluetooth becoming a standard feature on most handsets, there is an available free distribution channel on many mobile handsets. In 2006, it was estimated that there were a billion Bluetooth enabled devices worldwide with weekly shipments continuing at a pace of 12 million per week, according to the Bluetooth Special Interest Group[1] (SIG). The handsets are predominantly mobile phones. With many mobile handsets now having Bluetooth as a standard feature, this free channel can be leveraged to extend the PC Web-based content distribution scheme offered by the LMSs to mobile handsets. Using the Bluetooth distribution channel, we are enabling the distribution of educational content at no added cost to the students (cost being a hindrance to the SMS channel being used for LMS access).

RESEARCH OBJECTIVES

We carry out this research to achieve two major objectives

1. To explore the possibility of extending existing Open Source LMSs to distribute educational content to Bluetooth-enabled mobile handsets. As a case in point we prototype a distributed Content Distribution Architecture (CDA) modelled on Service-Oriented Architecture paradigm. This prototype extends one of the popular Open Source LMSs, Sakai, to distribute educational text content to handsets like phones and PDAs. It integrates the mobile handsets owned by students into an existing technology infrastructure at no additional cost to students.
2. The second objective is to evaluate the performance of the CDA in terms of content download response times. This will be done through experiments and it will help us in recommending the most suitable environment for its deployment.

---

1 http://www.bluetooth.com/Bluetooth/Press/SIG/BLUETOOTH_WIRELESS_TECHNOLOGY_SURPASSES_ONE_BILLION_DEVICES.htm

In this study we design, implement and evaluate a prototype system that distributes educational text content to mobile phones and PDAs through Bluetooth technology. Our design leverages resources offered by Sakai and uses an existing university network. We use a branded Sakai implementation called Vula that is customized for the University of Cape Town (UCT). UCT is an African University and serves as an example of a developing world university that uses an LMS in its learning activities.

The CDA design takes advantage of the extensibility of the LMS through Web services and is modelled on the SOA paradigm to take advantage of the already existing resources (LMS and mobile handsets) in UCT (which resources also exist in other similar developing world universities). Generally SOA is an architectural style for building systems based on interacting independent components called services. As a style, it is independent of the underlying technologies used to implement the services but its benefits can only be realized through implementing the services based on it. It is commonly implemented using Web services technology though other suitable technologies like Java Message Services (JMS) and Distributed Component Object Model (DCOM) can be used.

Since we are prototyping a service that we hope will be used on various Open Source LMSs (mostly implemented using various technologies, though offering a similar service like content distribution) it is imperative that our design integrates transparently with these systems. This integration must then use standards. Since most Open Source LMSs support extension of their functions using Web services (which are a standard that hides the implementation details of underlying services) our design is compelled to follow the SOA paradigm, whose implementation today is popularly realized using Web services.

Our CDA design consists of three independent modules (services) that interact during runtime. Bianco, Kotermanski and Merson (2007) describe a service as a distributed component whose implementation detail can be hidden. Services are the central pillar in SOA–modelled solutions and we inherently use them to implement our architecture. The system modules include;

- Web services module hosted by the LMS (Vula)
- Packaging module

- Distribution module

These services are distributed and their interaction only manifests itself at runtime. We use the runtime view as suggested by Clements et al. to best capture the conceptual design of the CDA as shown in Figure 1:1 (Clements et al., 2002). The figure shows the three main services during runtime interaction.

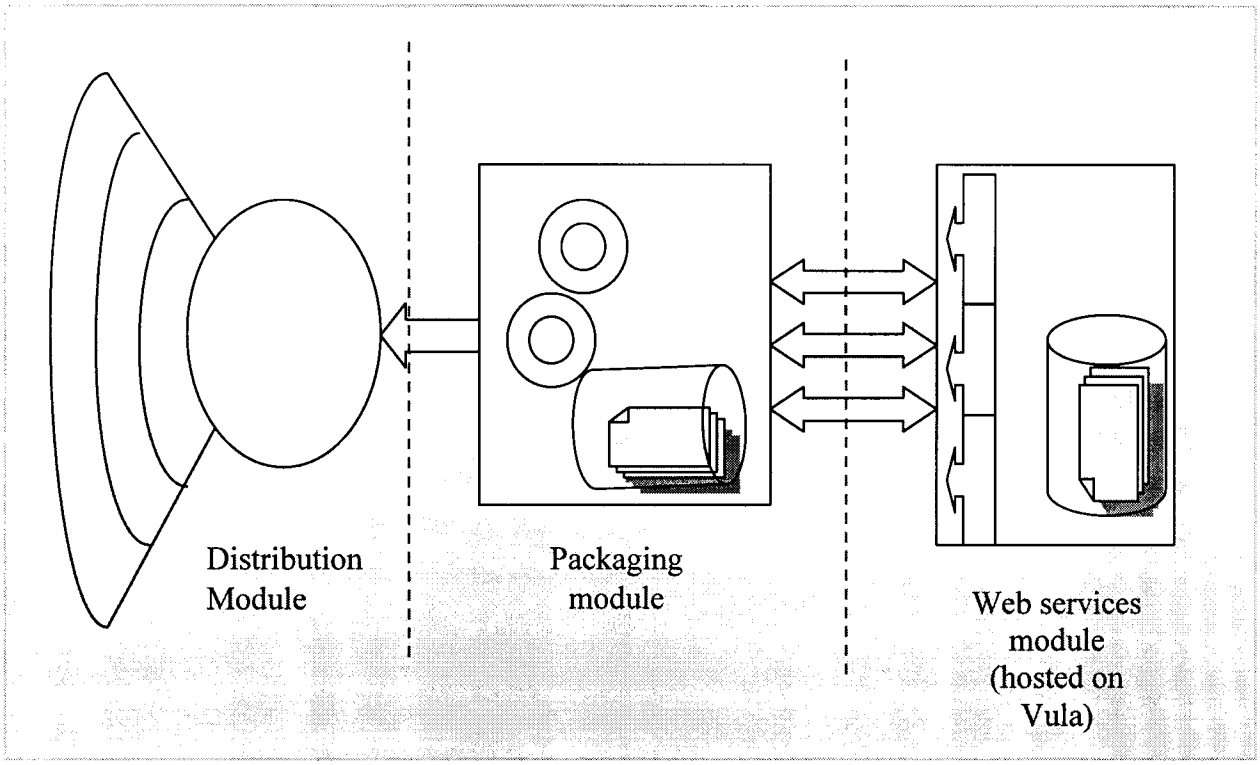


FIGURE 1:1

Runtime interaction of the system modules

METHODOLOGY PREVIEW

The extensible nature of most LMSs through Web services gives us a head start in developing a service that extends LMSs. We are designing a service that must be affordable (cost free in this case) and can be easily integrated into the daily routines of students (some of the key 'Real access' criteria for successful developing world ICT projects as laid out by Bridges.org[2] (an NGO looking at digital divide issues)). The existing and already familiar Bluetooth connectivity offers a free communication channel.

[2] www.bridges.org

After the design and implementation of the CDA, we use experimental evaluation (Preece, Rogers, & Sharp, 2002) to evaluate the system performance. According to Tutsch, performance evaluation covers different methods to determine system performance that include measurement, simulation and mathematical methods (Tutsch, 2006). We employ the measurement method since we are not evaluating the distribution service as a networked architecture (which may require using simulation and mathematical methods to determine attributes like throughput and latency) but as content distribution service (in which we are more interested in how responsive it is to the needs of individual students). During the experimental evaluation of our service, we determine how much time the distribution module takes to push files to a registered device during light and stress load situations.

In one of the experiments, we investigate how the number of text files to download affects the overall content download time for a user. We then use scatterplots to visualize the results obtained from this experiment and highlight this relationship. This relationship helps us in determining how the service degrades when a user is downloading a certain number of files and/or the time it takes to download a text document of a given size to a user's mobile device.
We then carry out two other experiments to benchmark the performance of the service. This performance is measured as system response time. We define system response time as the time taken from when a user activates their Bluetooth to the time when the service sends the first document if any, to the user's device. After the service responds, the user will know whether he/she has any documents or not. We envisage two use cases in which the service may respond differently depending on the usage load; light and stress load situations. We then design the two experiments to simulate the social context (Jones & Marsden, 2006) in which these situations are likely to occur. The results obtained from each experiment, are quantitatively analyzed using descriptive summaries that includes the mean service response time, and visualized using histograms to graphically show how the service responds in these situations. We finally compare the response time from both experiments to recommend a suitable deployment context for our service.

In this chapter we have introduced, motivated and proposed a system that we will subsequently prototype and evaluate. The aim of the system is to distribute educational text content to Bluetooth enabled mobile devices. It is modelled on a SOA paradigm and extends one of the most popular Open Source Learning Management Systems (Sakai) through Web services. Since most Open Source LMSS support web service extensibility (Aberdour, 2007), our architecture may be used as a blueprint to develop related educational content-distribution systems. In chapter 2, we review the underlying technology framework that we use to implement the proposed service, examine existing related Web-based LMSs that extend Sakai and Bluetooth content-distribution systems in commercial and non-commercial domains. In chapter 3, we lay a foundation for the research methodology we employ to scientifically evaluate the implemented architecture for performance. We introduce the high-level view of our system architecture and discuss the design choices for its implementation in chapter 4. Chapter 5 delves into the system implementation technologies and their tradeoff during inter-module communication. In chapter 6 we design usability experiments to benchmark our system for average response times during light usage and stress load usages, in addition to investigating the effect of the number of files on the content download time per device. We also analyze the data obtained from the usability experiments and use descriptive statistics to subsequently discuss the results and visualize them using histograms. In chapter 7, we review our original research objectives, discuss our findings and generalize the results from the experiments into a wider context. We also highlight the limitations of our system; make recommendations for the system deployment environment and finally point out areas for the future extension of our work.

2. LITERATURE REVIEW

INTRODUCTION

In the previous chapter, we motivated and proposed an educational content distribution system to mobile handsets that will extend a Learning Management System called Sakai. In this chapter, we lay the groundwork for the subsequent design and evaluation of the proposed architecture. In the next section, we give a synopsis of the Service Oriented Architecture paradigm previously introduced as the blueprint on which we base our design. We highlight the key characteristics that make it suitable for this service. Since SOA is a style, it has to be realized in some form, and in our design we realize it through using Web services technology. In this chapter, we give a summary of Web service technology and its key specifications that include XML (eXtensible Markup Language), SOAP, and WSDL (Web Services Description Language). In this section we also give an overview of Bluetooth as the content distribution technology in our service and highlight the specific features we adopt in the design and implementation. The section that follows reviews related architectures to those that are now implemented using the SOA model.

In the existing system section, we appraise systems that have extended existing resources in universities through Web services, systems that commercially distribute content using Bluetooth technology and those that use Bluetooth as an education content distribution medium. We highlight key characteristics of these systems, their underlying implementations, the targeted platforms and rationale behind some. We also highlight the model that we adapt in the analysis and evaluation of our service. Finally, we conclude the chapter by highlighting the common characteristics of the reviewed systems, and what they address, where the gap remains in as far as distribution of educational material and for which we propose a solution.

SERVICE ORIENTED ARCHITECTURES

We base our service design on the SOA paradigm. As already mentioned, SOA is an architecture style and not an implementation technology or solution. So what does this mean in relation to a system that has to be designed and realized as a technology artifact? By using the SOA model in our design, we are aiming at its inherent advantages that help to future proof our service design. But before delving into the details of this, we need to provide a clear definition of the term SOA that we will subsequently adopt to develop our arguments. There is no unified definition of the term SOA, but we adopt Erl's modern definition that captures the fundamental of most of the existing definitions. Erl describes the primitive form of SOA on the basis of it being supported by major vendor platforms, as

"a term that represents a model in which automation logic is decomposed into smaller, distinct units of logic. Collectively, these units comprise a larger piece of business automation logic. Individually, these units can be distributed" (Erl, 2005b).

He highlights that although distributing automation logic is nothing new, what makes a service-orientation separation unique is that it encourages these individual units of logic to conform to a common set of principles that allows them to evolve independently, while still maintaining a sufficient amount of commonality and standardization. In SOA, these individual units are known as services and we will interchangeably call them 'modules' throughout our discussion.

A service as the key building block in architectures modeled on SOA is defined by many authors. Barry defines a service as a well-defined, self-contained function that is independent of the context or state of other services (Barry, 2003). In the OGSA glossary of terms, a service is a software component participating in a SOA that provides functionality and/or participates in realizing one or more capabilities (Treadwell, 2005). In many related definitions of a service as a building block for SOA, it can be seen that they are treated as black boxes, that hide their implementation, but offer interfaces for visibility as they participate in service compositions.

SOA as an architectural style is implementation-agnostic and can be realized with any suitable technology; for example, Web services, component technologies like Common Object Request Broker Architecture (CORBA), messaging services like Java Messaging Service (JMS) etc. These architectures in following the same theme of implementation-technology independency

must still utilize services as basic elements to realize applications (Papazoglou & Georgakopoulos, 2003). Services as building blocks can be written in any programming language too.

However, as a style, SOA needs to be realized through an implementation which may use a suitable technology to leverage its advantages. The most widely used of these technologies, is Web services (Erl, 2005a). This popular realization of the SOA model through Web services evolves the term to a contemporary definition that encompasses its fundamental definition. Erl defines a contemporary SOA as one that has evolved the primitive definition, highlighting its key characteristics and that it is implemented using Web services (Erl, 2005b). This modern definition brings to the fore the advantages that accrue with using Web services.

- He also highlights the key driving principles in adopting the SOA paradigm in designing systems as (Erl, 2005a):

- Loose coupling-where services maintain a relationship that minimizes dependencies, and only requires service-to-service awareness
- Service Description in which well defined interfaces provide service functionality through service description documents which are adhered to by services during communication.
- Autonomy-services maintain control over logic they encapsulate.
- Abstraction-services hide their logic from the outside world and can be seen as black boxes whose functionality is only visible through interfaces. These interfaces act as service contracts that must be used for any external service that communicates with that service.
- Reusability-logic is separated into services with intention of promoting reuse.
- Composability-aggregation of services to assemble composite services and processes.
- Statelessness-services minimize retaining information specific to an activity.
- Discoverability-services are designed to be self-describing so that they can be found and accessed via discovery mechanisms.

As SOA is a relatively new paradigm there are bound to be contentions in the various standards that it uses from different standards bodies. As a result, some of the advantages can be best

realized through adoption of an optional standard. For example traditional SOAs follow the Publish-Find-Bind paradigm. In this paradigm services are published by the service provider in a registry, the service consumer searches the registry for the services and then finally binds to the service provider. In these architectures, use of the Universal Description Discovery and Integration (UDDI) as a service registry is emphasized but it does not have industry-wide acceptance and may therefore be left out as an optional standard in realization of contemporary SOAs. This could lead to systems being developed using the contemporary SOA definition to be subjectively viewed as SOAs depending on which definition you take, but fundamentally, they all conform to the primitive definition.

WEB SERVICES

In the definition of a modern SOA by Erl, he emphasizes that it can be implemented using Web services. However, since Web services is an implementation technology and SOA a style which is implementation independent, the two can exist independently, though they are popularly married to augment each other. The Web service technology has been initiated by industry and not academia and this is exemplified by the various evolving definitions from large companies and standards organizations (Steinmetz & Wehrle, 2005). One of the current definitions according to (W3C, 2004) reads as follows;

"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards." (W3C, 2004).

We adopt a more general definition from Srinivasan and Treadwell which defines Web services as distributed software components that provide information to applications through an interface (Srinivasan & Treadwell, 2005). This information is structured using XML so that it can be parsed and processed by other applications or services in this regard. Web services provide the service abstraction characteristic to systems modelled on the SOA paradigm by publishing the interfaces for services while maintaining their implementation details private. This implies that

services that support common communication protocols can interact regardless of the hosting platform or programming languages in which they are written (Srinivasan & Treadwell, 2005).

1. Web services as technologies that are widely used in distributed heterogeneous environments like architectures modelled on the SOA paradigm consist of three main specifications as outlined by Srinivasan & Treadwell.

2. XML: a markup language for formatting and exchanging structured data.
3. SOAP: an XML-based protocol for specifying envelope information, contents and processing information for a message.
4. WSDL: an XML-based language used to describe the attributes, interfaces and other properties of a Web service. A WSDL document may be read by an application to learn about the service.

Web services can support any communication protocol, but the most common is SOAP over either HTTP or HTTPS, our Web services use the former.

BLUETOOTH TECHNOLOGY

In our service, Bluetooth plays a very significant role as a protocol/channel that is used for the actual transfer of text documents from the distribution module to the Bluetooth enabled devices; for example, mobile phones and PDAs. Bluetooth is a low-power wireless Radio Frequency (RF) signal that operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4GHz to 2.485 GHz. It can penetrate physical obstacles like walls and does not have to be in line-of-sight like IrDA (Infrared Data Association, a standard for infrared communication). It uses a frequency hopping scheme to minimize signal interference caused by HomeRF, 1EEE 802.11 and other signals operating in that free band (Woodings, Joos, Clifton, & Knutson, 2002). According to Bluetooth SIG (2009), a Bluetooth channel can have different operating range depending on the device class. For example, Class 2 radios most commonly found in mobile devices have a range of 10 meters or 33 feet, Class 1 radios used primarily in industrial use cases have a range of 100 meters or 300 feet (Bluetooth SIG, 2009). It is capable of transmitting data at nearly 1MBps, though a practical rate is estimated to be 725Kbps (May, 2001).

For a device to connect to another Bluetooth device within its range, it has to discover the other device before a connection is set-up. This Bluetooth device discovery and connection procedure is, however, a time-intensive procedure that involves inquiry and paging sub-states. Device connection begins when the device enters an inquiry sub-state to discover other Bluetooth devices within its range. In the inquiry sub-state devices generate an inquiry hopping sequence (channel changing), that covers a 32-channel subset of the available 79 Bluetooth channels (Woodings et al., 2002). Here all the discoverable devices within the 10-meter range will respond to the device inquiry after which a user can manually select the desired Bluetooth device from the list of discovered devices. After the inquiry sub-state, the device enters a paging sub-state that allows it to establish a connection with the remote device. It uses the generated hopping sequence obtained from the inquiry sub-state. On completion of the paging sub-state, the devices move to the connection state.

The inquiry sub-state may have to last for 10.24 seconds according to the Bluetooth specification. In our architecture, the distribution module frequently searches for Bluetooth devices that are within its range. This implies that it at least spends 10.24 seconds to discover any registered devices within a 10 meter radius (this range is limited due to the use of class 2 radios most commonly found in mobile devices). This discovery time may in fact increase when devices are moving past each other, especially in an error-prone environment as *"there is no guarantee of successful inquiry even if both devices are on the same frequency at the same time, since packets transmitted at that time may be corrupted"* (Woodings et al., 2002)

In order for Bluetooth enabled devices to communicate effectively, they need to conform to the Bluetooth specification. The specification defines the standard a Bluetooth enabled device should adhere to, including the rules that need to be enforced during communication. It consists of Bluetooth protocol stack and profiles. The protocol stack is software that has direct access to the Bluetooth device. In the implementation of our distribution module, we target the OBEX (Object exchange) layer of the protocol stack. OBEX is used in the transfer of text files from the module to the devices. For our design we ignore the profiles since we are not developing any clients for the content consuming mobile devices. Bluetooth is inherently client-server architecture; the one that initiates the connection is the client, and the one that receives the connection is the server. In our design we take advantage of this intrinsic characteristic.

CLIENT-SERVER ARCHITECTURE

Systems that are modeled on SOA principles leverage its advantages through implementing services which are invoked by service consumers (client applications). Since we will implement our design using the SOA principles, we will also employ the service-service consumer model. In the past, there have been architectural models that have been used to allow for the distributions of applications physically and logically. Our service-service consumer model resonates with one of these earlier and evolving architecture called the client-server architecture. We review the client-server architecture with some of its evolved architectures in the following paragraphs.

The earliest of these architectures was the client-server. The client–server model still exhibits itself in just about any piece of software in which information is requested and received. Almost all the application architectures that exist have a variation of this model. The industry term "client-server architecture" generally refers to a generation of early environments in which the client and server played specific roles. The first client-server architectures consisted of 2-tier variation of the model and emerged in the late 80s. They evolved on the earlier monolithic mainframe systems where bulky back-ends served thin clients. The 2-tier design, introduced the concept of delegating logic and processing duties to individual machines, resulting in the birth of fat clients (Erl, 2005b). Common configurations of the architecture involved multiple fat clients doing most of the processing and presentation, and were connected to a centralized database.

Erl highlights that in 2-tier architectures, most of the application logic is placed in the client resulting in a large executable that controls the user interaction. The business rules are mostly embedded and maintained in the database in the form of stored procedures and triggers. Communication between the client and server is synchronous and requires each client to establish a persistent database connection. In comparison to SOA, processing is highly distributed and each service can ideally be deployed and run from any location, thus with no fixed processing. Communication between the services in SOA can be synchronous or asynchronous too.

Erl argues that SOAs can be viewed as distributed Internet architectures (Erl, 2005b). He highlights that earlier distributed Internet architectures could be designed as SOAs and some of the designs may have been heavily influenced by service-oriented principles, but traditional distributed Internet architectures differ slightly in the manner in which they were commonly designed. They are component-based and break up the monolithic client-based executable of 2-tier architectures into modules, giving rise to a multi-tier or, commonly called, the n-tier architecture. Application logic is distributed among the components. These components can reside on the client side or the server side. A single connection can manage multiple users, by allowing pooling of server side components to manage incoming database connections. These traditional n-tier architectures replaced client database connections with Remote Procedure Call (RPC) connections. They mainly used proprietary RPC technologies like CORBA and DCOM that managed communication between the components residing on the client and server side.

The arrival of the World Wide Web (WWW) as a computing technology in the mid-to-late 90s, led to n-tier architectures incorporating Internet technologies, resulting in distributed Internet architectures. Here a new physical tier called the Web server was introduced; HTTP replaced RPC calls as a communication protocol for client server communication, and limited RPC to enabling communication between remote Web and application servers. This architecture represented the de facto computing platform for the enterprise from late 90s to the mid 2000s (Erl, 2005b).

Distributed Internet architectures place their application logic on the server side thus centralizing their solutions since no logic exists on the client workstations. Executable client-side scripts are downloaded from the Web servers to respond to user events on the Web pages. These architectures are very similar to SOA; the difference only arising through the principles used to determine their primary design considerations as outlined below (Erl, 2005b).

- How application logic should be partitioned
- Where the partitioned units of processing logic should reside
- How the units of processing logic should interact.

We adopt the above SOA design considerations in the design of our architecture to model it using the SOA paradigm rather than merely creating a distributed Internet architecture. We discuss in detail these design considerations in the design chapter and the rationale for choosing them.

RELATED CONTENT DISTRIBUTION SYSTEMS

EDUCATIONAL CONTENT DISTRIBUTION SYSTEMS

In most universities, Learning Management Systems are used as holding repositories and distribution channels for electronic education content. These can co-exist with already established physical infrastructure like libraries. In these universities too, some of the physical libraries used for archiving hard copy books, journals and the like, have been digitized and today provide on-line accesses to this educational content for students.

The University of Michigan has developed such a system as a service that extends a digitized library and provides electronic library reserves to a Course Management System. This service is a locally branded implementation of Sakai, called CTools, and uses a customized RSS (Really Simple Syndication). RSS is an XML dialect primarily used to syndicate news content, repurposed in this service to deliver library reserves data. Dueber and Hollar (2006), in their report on the service, highlight that its goals were automation, reducing the work required by librarians and instructors, use of standard formats (to allow data to be used by many applications), and live updating (to accurately reflect the underlying live data at the library so that users can use the service with confidence).

The service is built on existing infrastructure that includes course information stored in the library catalog called CTools, and a registrar service that tracks (among other things) which students are registered for which courses and stores unique identifications for those courses. The service fulfills its initial design goals above through:

- Automation by relying on data available to CTools and the library system that tracks the reserve items
- Use of RSS for content distribution, a standard format that can easily be reused in a different service or transformed using standard tools

- And caching, implemented by the RSS Builder, for live updating and balancing of service efficiency and performance as desired by the users.

The CTools service report further highlights that data must be kept in a system that can be usefully programmed by providing programmatic access through interfaces and that each course must be uniquely identifiable. In addition, the Course Management System must be able to consume data from an outside source. In our lecture content distribution system, we use Vula (a University of Cape Town branded installation of Sakai) that provides some of the important infrastructure that is needed in the service. This infrastructure includes course identification, Web service hosting facilities, ability to upload content, an already existing user base (the students) and centralized database. In our service we extend Vula as an existing infrastructure in the University of Cape Town, which offers a Web-based educational content distribution service, to distribute this content to mobile devices through the use of Bluetooth technology.

There are other popular commercial CMSs like WebCT, Blackboard, FirstClass, ClassFronter, in use in universities though their extensibility and customization may be limited by their licenses and terms of use. We have thus limited the comparison of our CDA to CTools as a content distribution tool that uses a similar extensible open source LMS Sakai.

COMMERCIAL CONTENT DISTRIBUTION SYSTEMS

In educational institutions, however, Bluetooth technology has not been fully exploited for its potential as a distribution channel for educational content in the same way it has been commercially exploited for marketing and advertising. In this section we review some of the commercial applications of Bluetooth, by examining some of the proprietary systems that currently exist in the commercial domain.

A company called Filter UK has developed a commercial system for content distribution called BlueCasting that delivers text files, and multimedia files like audio, video clips and Java applications using Bluetooth (Filter UK, 2006). The system consists of a BlueCast server running a set of software services. This server identifies Bluetooth enabled handsets, tracks users and delivers customized messages to them. It is installed in retail locations, entertainment venues, public spaces, and interaction kiosks.

Our current content distribution architecture implementation will relate to the BlueCasting system in that it is intended to distribute text content to handsets using Bluetooth technology. The device interaction scheme for our system will also be similar to the BlueCasting systems' device interaction model in that, when a user is within its vicinity (discovery range), he/she must make the handset discoverable by switching it on (activation). The server will then identify the handset using its Bluetooth ID (a unique code that each device has which is like a MAC address or an IP address). The BlueCasting systems' content delivery approach is rule-based, and our architecture will use a similar approach, where a user handset will receive a new document that it has registered to receive only once. This will guarantee that only newly registered for and available content is sent to the user.

The two systems are intended to differ in implementation details; our distribution architecture will use Web services and its related specifications of SOAP and WSDL for content access and acquisition from the repository (content hosting module). It will also use XML to maintain and transform the original document structured format during the transfer of the text data to the distribution module. The use of these open communication standards like SOAP over HTTP and programming language-neutral data exchange through XML creates advantages that the LMSs can be hosted anywhere and accessed over the Internet or in a LAN, and the data can be consumed and /or transformed by different applications. Due to the extensibility of Web services, we expect our architecture to be easier to scale to accommodate growing demand. In the BlueCasting system, content is added and distributed from an instance of a server which has to be deployed at each location the system is to be used. This lack of ability to have a centralized repository hosted in one location and accessible via the Internet makes it a comparatively less scalable implementation than our system. For example, in our CDA, the extensible nature of the content hosting module (through Web services) means that more distribution modules and content packaging modules can be added at different locations to package and distribute content from the same central repository unlike in the BlueCasting system where such scalability can be achieved only by duplicating the content server when there is demand for similar content in a different location (since the server is not extensible via the internet).

In another Bluetooth broadcasting system called BroadTooth developed by Londondev, custom message management software is used to deliver advertisements and messages to Bluetooth enabled mobile phones and PDAs (Londondev Business Solution, 2006). Content is delivered

within a radius of 30 meters and includes multimedia files, graphic advertisements and animated GIFs. The system is mainly used in 'proximity marketing' and to send messages to high density audiences in places like sports events, music events or in rail terminals.

The BroadTooth system is managed from a single location, or via a LAN, and it is installed on the Windows Operating System (OS). Our architecture will be OS independent since all the modules are implemented in Java and can therefore be installed on Macintosh, Windows and/or Linux OS, provided they have a Java Virtual Machine (JVM).

The Jellingspot is another commercial marketing content distribution system developed by Midletsoft. It consists of a Jellingspot Data Server (JDS) and clients deployable on Bluetooth enabled devices. The JDS acts as a point server (a host of software applications for short-range wireless devices clustered around a specific point, whether mobile or stationary, and integrates the software, devices, and users into a cohesive unit (Midletsoft)) that distributes content to wireless devices. The content includes electronic coupons, informational text, audio, video, and corporate information. Jellingspot enabled devices receive the marketing information, which is used to buy goods, receive text updates about say new clothes in the store, upcoming sales and corporate information. Users can also leave feedback, complete electronic forms, and browse catalogs and menus using the clients (Midletsoft, 2006).

The Jellingspot system differs significantly to our architecture in the use of specifically developed client software that currently only runs on Symbian Smartphones. In our system, content is to be distributed to Bluetooth mobile devices regardless of their operating systems, and is rendered using existing clients without any customized client software or device configuration. In our system, text content will be pushed to registered devices by the distribution module (which may be deployed on any Bluetooth enabled hardware such as a PC or laptop with a JVM). There will also be no user interaction with the distribution module such as sending content from their handsets to the distribution module. In the Jellingspot system, the user searches for the point server and can send or receive content from it, while in our system the distribution module will search for the user handset and will only push the content to the user device.

The distribution module is the endpoint to our educational content distribution service. It pushes the packaged text content to registered Bluetooth devices. Its role in the content distribution service means that its efficiency, in terms of how fast it achieves this, will impact the service use in the envisaged context of universities. In view of this, we look at analyzing our service and evaluating the distribution module performance to determine how efficient it can be.

We base our analysis and evaluation on works related to evaluation of distributed systems, and/or peer-to-peer networks. In such a work by Theotokis and Diomidis, they propose a framework for analyzing peer-to-peer content distribution technologies (Theotokis & Diomidis, 2004). Their approach focuses on the system's non-functional characteristics of security, scalability, performance, fairness, and resource management potential – and the way these affect the architectural design decisions adopted by peer-to-peer systems. In the analysis and evaluation of our content distribution service, we deal with two of the non-functional characteristics borrowing from their approach, which include:

System scalability which involves maintaining system's performance attributes independently of the number of deployed distribution and packaging modules. These modules can be installed on different machines or in different physical locations like WANs or LANs. We theoretically analyze the number of communication calls the packaging module makes over the network to the Web services hosted by Vula. This will help us predict how the services' communication calls contribute to overall network traffic. Also, each content access and acquisition call from a deployed packaging module contributes to the total number of calls that can be concurrently handled by the LMSS (which is limited, hence limiting the number of deployed packaging modules that can access it concurrently).

System Performance which is the time required for performing the operations allowed by the system (Theotokis & Diomidis, 2004). In our service, we look at the time it takes the user to get a response from a deployed distribution module during light service use and peak service usage (stress loads). We term this "service response time". It includes the time the distribution module takes to scan for, and discover an activated Bluetooth device within its discovery range and finally make a connection to it, before starting to push content to it. We also examine how the

actual content download time (or content pushing time by the distribution module) is affected by the size or number of files that are sent to a device.

CONCLUSION

An overview of the reviewed education content and commercial content distribution systems shows that their content is either added locally to a content distribution server (in every place it is installed) and/or in a centralized LAN-based location. The commercial architectures mainly target specific operating systems for the distribution server installation or specific brands of Bluetooth devices. In some of the commercial systems, clients are specifically developed to be used for consuming the distributed content. In most of those systems, content is mainly distributed for marketing purposes and varies from multimedia to text. In related education content distribution systems, users are restricted to Web-based clients for content consumption. Our architecture specifically distributes text content and is meant to be used mainly in Universities and tertiary institutions to distribute content that is entirely text-based to Bluetooth enabled mobile devices without any custom made clients for them. The system is modeled on a SOA and designed to be scalable and installable on any operating system.

3. RESEARCH METHODOLOGY

INTRODUCTION

In previous chapters, we introduced our study through the introduction of a proposed lecture content distribution architecture and reviewed literature and related systems (both commercial and research systems). However, in scientific research, findings of studies have to be achieved using an acceptable scientific framework to allow for a structured approach to knowledge accumulation, to stretch the boundaries of what we know in a given domain (Whitman & Woszczynski, 2004). These frameworks allow the research findings to generate debate amongst the domain practitioners, and allow their generalization and application in wider contexts. This chapter presents various research methodologies, one of which we will use to ground our study of the proposed architecture.

The approach we take for the service design focuses on the SOA paradigm and its implementation using Web services, with the goal of extending an Open Source LMS to distribute course content to Bluetooth-enabled mobile devices like phones and PDAs in learning institutions. In the study, we create a functional prototype and use experimental evaluation methods to evaluate it (Jones & Marsden, 2006). Designing a functional SOA-modeled content distribution service will act as a proof of concept. We look at important descriptive qualities of SOA, methods that are appropriate for designing SOAs and their tradeoffs, and validation procedures required to establish knowledge claims that the content distribution architecture is indeed an SOA. After implementation, we test our service for performance using average and stress loads, present, and interpret our results, and finally conclude our study with recommendations of how to best deploy the system in a learning institution based on our findings.

Mouton (2001), defines a research plan as a blueprint documenting a particular research design. Research, is categorized as either qualitative or quantitative, the choice of which type and design substantially depending on how well each addresses the research questions in supporting the research objectives. In the following section, we give a brief overview of the quantitative research approach by highlighting the methods we will employ and why we pick it over a qualitative approach.

QUANTITATIVE APPROACH

Quantitative research aims to determine the relationship between a thing (an independent variable) and another (a dependent) variable in a population (Hopkins, 2000).
Hopkins further highlights that quantitative research designs are either descriptive (observational) or experimental (longitudinal or repeated measure).

- Descriptive or observational studies (once-off measurement of subjects), aim to establish associations between variables. Hopkins explains that no attempt is made to change behavior or conditions—things are measured as they are and accurate estimates of relationships between variables are obtained using hundreds or thousands of subjects.
- In experimental studies, measurements are taken, some intervention is tried, and then measurements taken again to see what happen after the intervention i.e., subjects are measured before and after to establish causality. Estimates may be obtained with tens of subjects. Experimental studies are also known as longitudinal or repeated measure studies.

However, in both quantitative research designs (descriptive and experimental), the final results may be biased. This can be reduced by randomly selecting subjects, in addition to the subjects and researcher being blind to identity of treatment in experiments. In descriptive studies, having a high participation rate in selecting random samples from the population, makes the estimate of the relationship less likely to be biased. In quantitative studies, however, subject characteristics can affect the relationship between the variables (dependent and independent) but this effect can

be limited by using a less heterogeneous sample of subjects by preferably including the characteristics after measuring them (Hopkins, 2000).

In our study of the SOA-modeled content distribution architecture, we structure our study as a quantitative study to measure key non-functional attributes like effect of content size on download time, and service response time during average load and peak load service usage. These benchmarks will help us in planning service scalability and determining a suitable service deployment context within learning institutions. We chose a repeated measure quantitative study due to the following reasons

1. We have a limited time and a limited number of phones and PDAs to carry out the experiments for the study using a very large representative sample size of university users.
2. We are designing and implementing a lecture content distribution architecture and want to analyze its performance. This will give as an indication of how efficient and scalable the service can be when deployed in various contexts. It will also highlight performance bottlenecks that should be optimized to achieve practical service usability, through, say, reducing service response time.
3. We aim to investigate service response time during envisaged peak and average service use scenarios in learning institutions, which is a perfect case for cause and effect experiments.

The above study goals and limitations make a qualitative study unsuitable and thus we choose a quantitative approach. Due to the time constraints of the study, we use the simplest form of experimental studies; time series. In time series experiments one or more measurements are taken on all subjects before and after treatment (Hopkins, 2000). However, in these experiments, there is no control group. In one experiment, we will use a special case of time series called the single-subject design (measurements are taken repeatedly e.g., 10 times) before and after an intervention on one or a few subjects (Hopkins, 2000). This experiment will be an investigation of how the number or size of pushed files affects the content download time to a device. Details of this experiment and its results are discussed in Chapter 6.

Time series as an experimental design, however, suffers from a major problem; any change you see could be due to something other than the intervention (treatment). For example, subjects

might have gained some experience from their first test, or the device you are using could have cached some data for subsequent content download sessions. To mitigate this effect in our experiment, we switch off the device and switch it on again before we push more files to it in the next content download session. Since the service distributes text files only, we remove the possibility that the devices employ different processing capabilities if the files downloaded were in different formats. In all our experiments, we also use the same type of devices (Windows PDAs) and uniformly configure their Bluetooth, implying that the same operation configurations are employed for each. However, the results obtained from these similar devices in the above mentioned experiment are just service performance indicators and are used to investigate the general relationships between the number/size of files and the content download time.

Hopkins highlights that the quality of evidence provided by the various quantitative designs (Hopkins, 2000) for the cause-and-effect relationship between variables differs. Cases and case-series of descriptive studies are the weakest, while experimental studies provide the best evidence about how something affects something else. Since we use the experimental study and not the descriptive study, we are confident of revealing cause-and-effect relationships that are scientifically sound.

MEASUREMENTS

SUBJECT CHARACTERISTICS

During any study we measure the characteristics of the subjects, the independent and the dependent variables defining the research question (Hopkins, 2000). For experiments, measurements can also be taken for mechanism variables to explain how the treatment works. In our experimental studies we identify our subjects as Bluetooth enabled mobile phones and PDAs which we occasionally refer to as devices. Using the Least Common denominator design approach suggested by Ballard (2007), we identify text as a global document format that can be rendered with the least degradation by most of these devices. Thus our system is designed to package and distribute text files to the devices. The devices we target are on the lower end of the device spectrum with less advanced document rendering capabilities (most do not support proprietary document formats like PDFs, Microsoft word documents, etc). These devices also have small display screens, are memory constrained and with short battery life. They are,

however, representative of devices held by students in learning institutions. We consider the device users and their profiles as inconsequential for our study since we are evaluating the educational content distribution architecture as a service. The scope of our study is limited to how efficiently the implemented system distributes the lecture material (in average load and stress load situations) and not how the distributed content is used for educational purposes.

RESEARCH QUESTIONS

In benchmarking the performance of the service, we identify content download time as one of the most important non-functional characteristic that makes the service practically usable within the university deployment contexts we envisage (e.g. cafeterias, libraries, campus bus stops, and near common and departmental notice boards). We define content download time as the amount of time a discovered device takes to successfully download a text file from the distribution module. It is measured from the time the distribution module retrieves the text file from its local file repository to the time it is sent to the device. Every registered device is eligible to download all the available content it has registered for, from a deployed distribution module when its Bluetooth is activated within a discovery range of 10 meters. To successfully download files, a Bluetooth connection has to be established between the distribution module and the device. Device connection authorization may be requested by the device but this depends on the individual device configuration (we configure the devices in our experiments for only one connection authorization per download session).

In the first experiment we investigate how content download time increases with the number of subscribed-to text files pushed to a device by the distribution module. We investigate this effect by formulating the following research question:

- How does content download time increase as the number/size of text files pushed to a device increases?

The above question leads to the formulation of a hypothesis that there is a relationship between the number of files pushed to the device and the time the device takes to download them. To investigate the relationship, we set up the first experiment to prove the null hypothesis (that there is no relationship between download time and the number/size of files pushed to a device) wrong

and thus confirm that the relationship exists. We then analyze the relationship and its effect on the overall service usability.

In the next experiment, we benchmark the average service response time for peak (stress load) and off-peak (average load) service use scenarios. For this experiment, we define service response time as the amount of time a user waits to get the first response from a distribution point when they activate their Bluetooth connections within a 10 meter radius. This service response may be a simple request for a Bluetooth connection from the distribution module after discovering the device (for devices that require connection authorization) or it may be a text file pushed to a device from the distribution module (for devices that do not require Bluetooth connection authorization). To benchmark the service we raise the following research question;

- What is the average service response time in peak usage (stress load situation) and off-peak usage (average load situation)?

To answer the research question, we formulate a hypothesis that service response time will be different in stress and average usage situations. We therefore design two service usage experiments to simulate the two usage contexts and disprove the null hypothesis that service response time will be the same in both contexts.

Service response time is an important non-functional aspect that will affect service usage in practical settings. A short response time may make the service more usable by time conscious users and in busy environments and will mean we can deploy fewer distribution points. Longer service response times could lead to user frustration with the service and may make the service less suitable for deployment in dynamic and overpopulated environments. It could also mean that if we are to deploy distribution points in busy environments, then we need more points to cater for the large numbers. Service response time in our architecture implementation is influenced mainly by the amount of time the distribution module takes to discover and establish a connection with a device within its vicinity. The overall user waiting time is also prolonged by how many files the user has to download after the service has responded to him/her.

Since the distribution service uses Bluetooth technology then the service response time is directly affected by the Bluetooth Device Discovery and connection time. The Bluetooth specification states that "*the inquiry sub-state may have to last for 10.24 seconds unless the inquirer collects enough responses and determines to abort the inquiry sub-state earlier*" (The

Bluetooth Special Interest Group, 1999). A user spending 10.24 seconds or more for the distribution service to just discover his/her device in its range may be an acceptable delay provided that the content is sent to the device as soon as it is discovered and depending on whether he/she is in a 'push and shove' or a relaxed/static environment. Device discovery time may even be longer (when devices are actively moving relative to each other).

Since content download time for each device may vary with the number/size of files and the device operation settings (is the device configured for Bluetooth connection authorization or not?), we will use similar devices but exclude the content download time when measuring the service response time (as per the definition of service response time in this context). The experiment will benchmark the average service response time during peak (stress load) and off-peak (average load) service usage contexts. The experiment simulates peak and off-peak service usage times.

We envisage a peak/stress load period as a busy distribution service usage period. The users are relatively mobile and are activating their Bluetooth connections at almost the same time. There are assumed to be many, already active devices are within a 10 meter range of the distribution service, waiting to be discovered and sent content. In this scenario, students immediately after a lecture are downloading references to reading material that they have not covered during the lecture, experiment results from previous laboratory session, test marks and answers, essay answers etc. The students are also time conscious, and want to download the material as fast as possible before moving to the next lecture. They are, as a consequence, moving past the distribution service (point server), or are relatively mobile, such as through a corridor to the next lecture venue.

During off-peak/average load service usage, active registered devices within a 10 meter radius of a content distribution point are few and users are more static. These devices are switched on one at a time within a reasonable interval of one another. Here, the students may be seated in a library, or standing at a bus stop or having lunch. They are more patient and are willing to wait for the content to be downloaded as they do some of the other activities i.e. they are multitasking.

By measuring and benchmarking service response time during these simulated usage scenarios, we aim to determine the most suitable usage context (peak or off-peak) in which distribution modules can be deployed to efficiently distribute content. Longer service response times will mean that the service should be deployed for average load usage as explained above; otherwise it

can be suitably deployed in both peak and off-peak contexts. It can also mean that more distribution points need to be installed in case the service is to be deployed in busy environments to cater for large numbers, since average system response time is long.

DEVICE SAMPLING

Due to a limited time frame for our research and limited monetary resources, we are unable to use an optimum sample size representing all Bluetooth enabled mobile devices in testing our prototype architecture. We therefore carry out a pilot for an anticipated larger University-wide deployment study of the education content distribution service. Alreck and Settle (1995) describe a pilot study as a brief preliminary survey, often using a small, convenience sample (Alreck & Settle, 1995). In our pilot studies, we first investigate the effect of file number/size on content download time per device and, secondly, determine the average service response during simulated peak and off-peak service usage. For the first experiment, we use the 10 observations per session to project values for a larger study for each experiment. In the second experiment each run is independent of the others, so we carry out a number of runs and chose the 10 most consistent ones (runs with minimal time differences between successive device responses) for analysis.

Since some mobile devices, especially phones, held by the students in University of Cape Town do not have Bluetooth technology (some devices only have Infrared and GSM connectivity alone), there is a likely device selection bias which is introduced by the narrowing of our sample device population to only the Bluetooth enabled ones. Since we are benchmarking the performance of the service and not the suitability of use of the distributed education content for learning purposes, this device selection bias is trivial to the main goal of evaluating the service performance. We, however, assume that the proliferation of Bluetooth enabled mobile devices, especially phones, amongst students is high enough for the distribution architecture to be deployed and used on the University of Cape Town and other learning institutions in the developing world. We also foresee that, since many devices are now shipped with Bluetooth technology, those without them will acquire them in the near future so that they can be able to use the service.

For these experiments, we use Bluetooth enabled Windows mobile PDAs (Pocket PCs) to represent the array of devices used by the students in a learning environment, taking the

University of Cape Town as a sample institution. This university, as mentioned, has an existing Open Source Learning Management System called Vula, which is a branded version of Sakai. Our prototype lecture content distribution system is designed, implemented, deployed and tested on the existing network infrastructure at the University of Cape Town. We deploy the distribution module on a Bluetooth enabled PC to act as a content distribution point and deploy the packaging module on the same PC. Our LMSS is hosted by another machine hosted on the same network as the other modules. We deploy and test all the three modules of our architecture as a system on these two Windows XP machines.

4. SYSTEM ARCHITECTURAL DESIGN

INTRODUCTION

There exists a wide range of mobile phone and PDA models. The variety presents a challenge in designing and implementing a "one technology fits all" solution, our education content distribution service being such an attempt. Design approaches, however, exist to counter this challenge: for example, Ballard highlights existing approaches for handling mobile phone proliferation as: targeted, least common denominator, automatic translation and class based (Ballard, 2007). In our work we are motivated by the least denominator approach that advocates for a minimalistic approach (selection of technologies and designs that will work on all devices). We copy this approach by identifying the set of features and capabilities that almost all mobile phones and PDAs support. We identify Bluetooth connectivity and text display as the basic feature and capability for the mobile device to consume and render text-based education content that the service distributes. In our design, we ignore advanced proprietary document rendering capabilities on some of the newer phones, smart phones and PDA models as devices with this capability are less common with students.

DESIGN DECISIONS

In any design, decisions about the specific characteristics the system is supposed to embody so as to achieve its designated overall goals have to be made. In his definition of a design characteristic, Erl (2005) argues that it has to be a specific attribute or quality of a body of solution logic, that is documented in a design specification and is planned to be realized in development (Erl, 2005b).

Our content distribution system uses a modularized service model that is deployable over a network (LAN, Internet) and can be hosted by any of the major operating systems (Windows, Mac OS, and Linux) running the Java Virtual Machine (JVM). The modules are designed to:

- Leverage some of the advantages of loose coupling like increased flexibility through the use of Web services, asynchronous communication, dynamic/static service binding, operating system and programming language independence and easy composition into high-value services.

- Repackage uploaded structured text educational documents into smaller documents to be pushed to Bluetooth enabled mobile devices.
- Communicate using synchronous and asynchronous schemes while employing standards like SOAP over HTTP, XML for interoperability and Bluetooth wireless technology for device and service connectivity.

MOBILE CONTENT FORMAT

SOA as an architectural style promotes a content and intelligence-heavy messaging model by using the document-style SOAP messages during client and Web service communication. This model supports service statelessness and autonomy and minimizes the frequency of message transmissions (Erl, 2005b). The intelligence-heavy messaging model is copied and implemented for content transfer throughout the three service modules in our architecture to borrow the service design principles of the SOA paradigm. This design decision helps in minimizing dependences within the three modules (distribution, packaging, Web services) and minimizes communication calls over the network on which the system may be deployed.

We use an XML format to package the text content as it is transformed and moved from the LMSS to the distribution modules. XML is key to the platform neutral data exchange between the modules. It provides a mechanism for creating complex data structures like DOM (Document Object Model) documents. For example, in the packaging module, the large structured text files acquired from Vula are transformed into DOM documents that are parsed for a desired preconfigured section like the abstract, conclusion, discussion, references, or acknowledgements. This section is extracted and its size is determined by the number of text lines preconfigured as an attribute in an XML file read by the packaging module at start-up. A sample structured text document uploaded in the LMS is attached in Appendix A and an XML extraction of its abstract section is in Appendix B. Figure 4:1 shows a sample XML transformation of the abstract of the same text file creating a document from the first ten lines.

```
<?xml version="1.0" encoding="UTF-8" ?>
        <documentlet>
                <siteName>CSC500W</siteName>
        <title>Supporting cooperative teamwork: information, action and</title>
        <line>This paper provides details of an in-depth investigation into</line>
        <line>how racing sailors use information displays and devices, and</line>
        <line>shows that these devices act as communication loci and</line>
        <line>instigators of action. The paper presents a detailed look
athow</line>
        <line>sailors use instrumentation on their boats for both their own</line>
        <line>performance and as the foci for developing a shared</line>
        <line>understanding: this is a detailed study of computer-
supported</line>
        <line>cooperative work in a new environment. We present a brief</line>
<line>summary of the ways that technology has pervaded the environs</line>
        <line>of sailing yachts, and analyze how this has affected the</line>
        <line>activities of the crew and altered the relationship between the</line>
        <line>sailors and their environment. We introduce a taxonomy of</line>
        <id>/group/csc400w2008/cooperativeTeamwork.txt</id>
        <creationDate>20070626101306234</creationDate>
        <modificationDate>20070626101306234</modificationDate>
</documentlet>
```

FIGURE 4:1

A Sample XML representation of a structured text document abstract.

DEVICE REGISTRATION SCHEME

During a content download session to a device, the distribution service pushes text files to the device after discovering and connecting to it within a maximum range of 10 meters. Device registration is done manually by adding a device Bluetooth ID to a device registration file in an XML format, shown in Appendix C. The ID is added to different XML files, named after the course/site from which a user wants to get content updates. The registration files are periodically parsed by the distribution module to extract the newly added device IDs.

A copy of the text file equivalent of the XML-format registration file is created and indexed by the distribution module, to assist in subsequent quick searches. The text file is saved in a local

folder that is written to and read by the distribution service. This local folder acts as a local repository for the distribution service and also contains the text documents that are to be sent to the registered devices. All files in this local repository are periodically indexed by a routine in the distribution service for quick searching. This periodic indexing is done because text documents are constantly being added and removed. Also, the distribution module has to be updated with a fresh list when new devices are registered with it. For every distribution module deployed, a local repository is created for holding text documents and registration lists. This localization of registration lists implies that each deployed distribution module is independent and can thus register devices and send content to them without knowing about the other deployed modules. The autonomy of the distribution modules thus makes our design easily scalable i.e. many distribution modules can be deployed at different places, installed on different operating systems and users can register to receive content from one or more distribution points, etc.

DEVICE REGISTRATION AND VERIFICATION

Before content is pushed to an active device (whose Bluetooth connectivity has been activated within the distribution module vicinity), the distribution module searches for the device's Bluetooth address. This search is done through the inverted index that is created for all the text documents in the distribution module's local repository. The indexed documents include registration lists and text documents awaiting distribution. This search through the inverted index is done by the indexing and searching routine we implemented in the design of the distribution module. The search returns all the names of the courses/sites for which the device is registered. If this search returns no course name, then the active device is not registered with the distribution module and hence no content is pushed to it, otherwise it is registered. A second search is done through the index that returns all the files names of the content the device is meant to receive. These files are then retrieved from the local repository and then pushed to the registered device using Bluetooth. This is a two-phase transaction that is carried out before any content is sent to any registered device. The two phases include:

1. Verify the device course/site registration
2. Acquire documents registered for by the device

For unregistered devices, the verification phase returns no record of registration and thus the device is ignored even though its Bluetooth connection is active.

TRACKING DOWNLOADED CONTENT

One of the key design goals of the distribution architecture is to only send registered users only new content uploaded in the LMSS to their mobile devices. This implies that keeping track of what documents the users have already downloaded over time is a key design issue. Since each distribution point may be accessed by different users (where each user might be registered for different courses), keeping track of the content they download means that users are sent only the content they have not yet received i.e. only new content. This will ensure that the service meets one of its key design goals of distributing only new content to the users. This will also ensure that users who have no content to download or have previously downloaded the content do not congest a particular distribution point (thus guaranteeing fair use of the service at the expense of duplicate downloading).

In the distribution module design, we ensure that users only get content they have subscribed for and only content that they have not received before (if available). We achieve this design goal by creating a download profile for each registered device the first time content is download to it. The profile consists of a Bluetooth address and currently existing files associated with it, together with the device's previous file download status. This profile is written to a file on the machine the distribution service is deployed on. When a file is sent to a registered device, its download status relative to that device is set to true, meaning the device has successfully received the, file otherwise it is set to false. This is done for each file downloaded by the device. When the download status is flagged true for all files, then the device has been sent all the currently available documents to which it is subscribed. This means that no more new content is available for it. For all the new content, the download status is set to false (to mark availability for sending) for the registered device, and is flagged true (already sent), after a successful download. The device download profile is serialized to a persistent store like the hard disk and de-serialized each time the registered device is discovered and a successful Bluetooth connection established, thus ensuring one download per file per registered device.

Content deletion and addition are basic operations that are used to refresh local document repositories in the distribution module and packaging module of the system. These repositories contain XML and text documents. Content addition and deletion is done to reflect the same operations of file upload and removal in the LMS course sites. In the design, we implement a cascading document addition and deletion scheme. Here, as documents are added and deleted from the course sites in the LMS, the same operations are carried out in the packaging module's local repositories, then the shared repository (between distribution module and packaging module which can be local or on a networked machine, in our case we use a local version) and finally the distribution module's local repository. These repositories and their contents are:

- LMS resource folders: Each course site has resource folders that may hold text and other multimedia documents that are uploaded or deleted by the administrator or site owners using a Web-based interface.
- Packaging module repository: It is a local folder where the copies of text documents acquired from the LMS are held. These documents are also appended with important metadata like resource IDs, date of creation and date of modification, a sample of which is shown in Appendix A.
- Shared repository: It is a folder that holds the XML documents that have been created from the structured text documents that are held in the packaging module repository. These XML documents are excerpts from a section of interest (size and this section of interest are pre-configured at packaging module start-up time) from the structured text document. These XML documents are appended (using XML tags) with the metadata from the large structured text documents. This shared folder can be hosted on a network, or locally. We use the latter case in our prototype.
- Distribution module repository: This contains both the text files created from the XML documents held in the shared repository above and the device registration text files, containing Bluetooth addresses of the device.

The shared repository is accessible by both the packaging module and distribution module. The packaging module only writes to it, while distribution module has read access only. We make

this module's access privileges mutually exclusive to prevent file corruption and maintain document integrity (for example, both modules cannot attempt to write to one file).

During the periodic calling of the Web services by the packaging module, only text documents that are present in the LMS sites are acquired. This implies that, when content is deleted from any LMS site resource folder, the acquired documents just after invocation might be different from the documents stored in the packaging module's local repository; just before that content acquisition call was made. In our implementation, the freshly acquired document file names are put in a list and compared with names on a list containing existing document names in the local repository. Any document that does not have its name in the fresh list is deemed stale and thus deleted with its corresponding extracted XML document from the shared repository.

Uploading of new text documents to a LMS course sites means there are newer resources in those sites in addition to those already existing. When the packaging module invokes a content acquisition call, the new content is retrieved in addition to the already existing documents from those sites. A fresh list of names of the acquired documents is returned from the call. This list is compared to the older list as in the content deletion context above. For names that appear on the new list and do not appear in the old one, it means that those documents are new. They are added to the documents in the packaging module's local repository, XML files are created and added to the shared repository from where they are acquired and parsed by the distribution module, thus updating the whole array of repositories.

The distribution module periodically reads and parses XML documents from the shared repository to create text documents to distribute to the mobile devices using Bluetooth. This ensures that addition and deletion of documents in the LMS course sites is propagated from the LMS to the distribution module, thus properly reflecting the underlying state of the documents in the LMS. This document updating model achieves a key user goal of availing only the updated content from the course site in the LMS to the registered users.

Our addition and deletion process though calls for the protection of the local and shared repositories from third party (external to the system modules) additions and deletions of content to maintain the data integrity throughout the process flow. This is left as a security detail of the service that is beyond the scope of current architecture implementation.

The distribution module pushes documents to devices that are within its range. Before document pushing, the most frequently done operation, apart from device discovery, is device verification and document searching. As explained, the discovery-verification process is done for each registered device and it involves searching through stored text documents (registration lists and the actual documents to be pushed). The discovered Bluetooth address is searched for in the registration files to ascertain which courses/sites the device is registered for. Having obtained the site/course names, another search is made for documents that have been obtained from the course sites in the second step. These documents are then retrieved from the local repository to be pushed to the device.

The two-step search procedure is a time expensive operation that can degrade the service response during heavy usage. For example, if there are many users where each is downloading many files, then the service delay will be significant because of the long searches. Optimizing this two-step search routine is critical to creating an efficient distribution module. A sequential search can be done for documents by looking for all the documents that have their course site as the site for which the device is registered. This sequential search is done for every registered active device. With a few documents present in the local repository, the system response time may seem reasonable, but with very large document sets, say a million documents, the two-phase sequential search will make the module response unacceptable.

In the design and implementation of the distribution module, we use an inverted index data structure. This structure is optimized for retrieval (search), update being of secondary concern. In this structure, text is inverted so that instead of the view obtained from scanning documents where a document is found and then its terms (list of documents each pointing to a term it contains) are seen, an index that maps terms to documents is built (like an index at the back of a book). The advantage of an inverted index is an increase in the speed and efficiency of searches in a document collection. An inverted index is optimized to do many accesses in $O(1)$ time, though updates may take a significantly longer time, being $O(n)$ in the worst case. Index construction time is longer as well, but query time is generally faster than a B-tree.

An inverted index data structure is commonly used in search engine domains, where data are searched for more frequently than they are updated. In the distribution module, a device's registration is verified by searching through the list of available course lists in the local

repository and the content to be pushed is searched for, from the document text files that exist in the same repository. The frequency of the search operations warrants an efficient search data structure in that more search operations are done than content updates which may be once or twice a day (our system is not a news or notification service).

We implement the inverted index algorithm by using a Java-based Open Source search engine library called Lucene[6] in the distribution module. It consists of core APIs that allow indexing and searching of text. All the documents in the local repository for the distribution module are indexed periodically to create a fresh index that includes the new content and updated device registration files. Discovered devices are verified by searching this inverted index from which a list of course sites is returned. Each course site name is then searched against this index to obtain a list of file names for the content that was originally acquired from the sites. These files are then acquired from the local repository and pushed to the registered device using a Bluetooth connection.

MODULE DATA SYNCHRONIZATION

The frequency with which the packaging module polls the content hosting service and the frequency with which the inverted index is refreshed by the distribution module are configured at startup time for both modules. Both modules read one XML file in which time for these polling intervals are set. Index refreshing time is configured such that it is almost in harmony with Web service polling time. For example, if the packaging module accesses and creates documents after every 12 hours, then Index refreshment time can be configured to take place every 13 hours. This is used because the service practically only needs to create a fresh index (to reflect the updates) when there is new content or when content has been deleted from the Learning Management System.

In our implementation, however, it also means that new devices added to the service can use the service only after the first index refreshment after their registration has taken place. Since search frequency in the distribution service is very high, and index creation frequency is very low in comparison, shorter searching times at the expense of lengthier index construction times is an appropriate tradeoff in our architecture implementation.

[6] http://lucene.apache.org/java/docs

5. MODULE IMPLEMENTATION

WEB SERVICES MODULE

Erl (2005) describes Web services as a Web-based distributed technology that leverages the concept of a standard communication framework, to bridge the disparity that exists between and within organizations. Web services primarily comprise a public interface; which is information that identifies the service and enables its invocation. This information specification is called the Web Service Description Language (WSDL). In our Web service design we predominantly use the WSDL specification documents. Web services use SOAP (originally Simple Object Access Protocol, but technically no longer an acronym (Srinivasan & Treadwell, 2005)) specification and other alternatives like XML-RPC as Internet-friendly and XML-compliant communications (Erl, 2005b), though currently SOAP has the widest industry use for Web service messaging. The latest versions of the SOAP specifications released by the World Wide Web Consortium [7](W3C) allow RPC-style and Document-style SOAP bindings for Web service invocations, but the latter is the most frequently used within SOA implementations.

Web service interaction occurs through passing XML data, with data types specified using XML schema. SOAP is an XML-based lightweight protocol that is used for encoding and exchanging data between applications (Steinmetz & Wehrle, 2005). It can be used over other communication protocols like HTTP, HTTPS, SMTP, FTP, UDP but SOAP over HTTP communications is the most prevalent use in exchange of SOAP messages between Web services and other applications. The input/output signatures for Web services are given by the WSDL. In our Web services, the WSDL is used as a service description for the public methods that are invoked by the packaging module. Service descriptions can be thought of as endpoints, through which messages can be sent and received. Hull and Su (2005) explain that WSDL specifies a "reactive" operation in which services receive a message. These reactive operations can be declared "one-way" in which case, there is no return response otherwise it is a "request-response" operation where a return type is declared. There are also "pro-active" operations that send messages from the service for example "notification" operations that send out messages without waiting for a response (asynchronous operations), and "solicit-response" operations that block and wait for a response

[7] http://www.w3.org/TR/SOAP

(synchronous operations), with the response type being specified with the operation. A service can be viewed as a server (via its reactive operations) and a client (via its pro-active operations) (Hull & Su, 2005).

The most significant attribute of Web services is their loose coupling. They are also independent of the infrastructure they are running on; i.e., the same service can be run on many different operating systems and hardware configurations with the same results on each of them, unlike component-based architectures which seemingly offer similar advantages to Web services. In our architecture, we leverage the advantages of loose-coupling and Web services communication using SOAP as illustrated in Figure 5:1, which shows the Web services module hosted on the LMS.

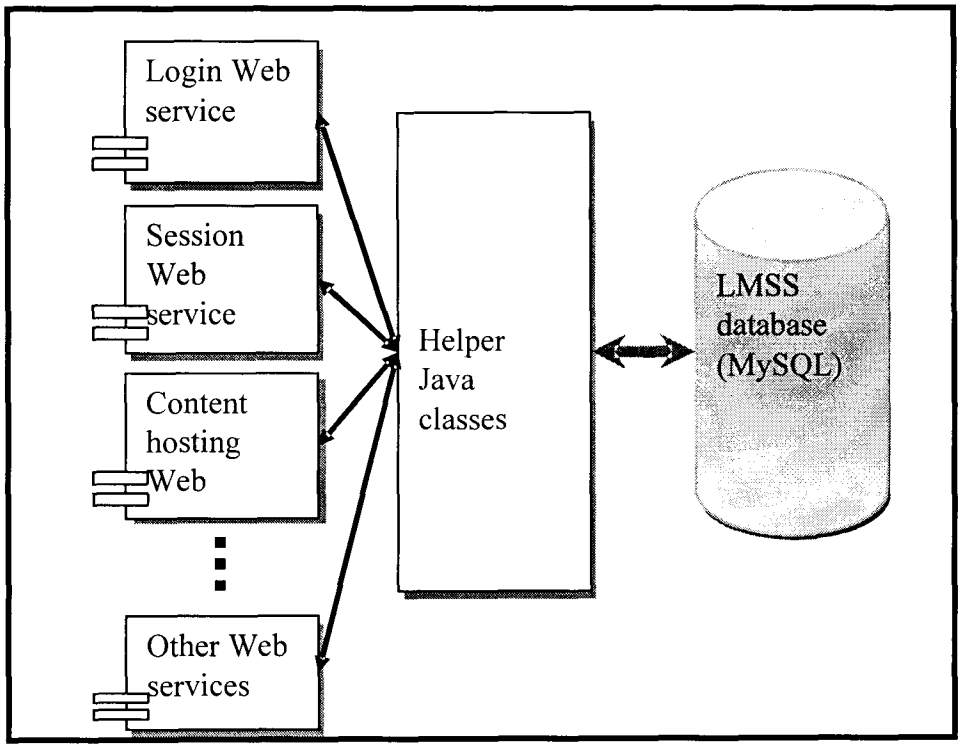


FIGURE 5:1

Main Web services and document repository in Vula (LMS).

Web services as technology components are hosted on a Web services platform, which is a set of tools for invoking and deploying them using a particular programming language. In our implementation the Web services are deployed on an Apache Axis Web services platform that is integrated with Vula. Vula is the LMS which we use and has a centralized database. It is a customized Open Source LMS, called Sakai, which is implemented using Java. It is used to upload and host education content for course sites in resource folders for the University of Cape Town. The Apache Axis Web services platform like many others, provides three core subsystems to the Web services; invocation, serialization and deployment. Hansen gives a detailed discussion of these subsystems (Hansen, 2007), we summarize serialization and deployment but elaborate the invocation subsystem to highlight one of the most important aspects of our Web service communication model. Hansen, summarizes the importance of serialization subsystem in Java Web services as

"... central to the process of invoking a Web service via an interface by translating the parameters (passed to the interface proxy) from instances of their respective Java classes into instances of target XML schema."

The deployment subsystem supplies the tools for setting up a Java target so that it can be invoked as a Web service via SOAP messages. It is responsible for publishing the WSDL document. This deployment subsystem as part of the Web services platform is integrated in the LMSs (Vula) to which, as one of its many functions, it makes the WSDL document available to the packaging module (which is the Web service client in our architecture) as a URL.

The invocation subsystem carries out the invocation process of the Web services by the clients and vice-versa. It is a complex process that can support SOAP or other alternatives like REST (Representational State Transfer). For example, it receives a SOAP message from a transport like HTTP, determines the message WSDL operation to invoke and then determines the Java class/method to invoke (dispatching) among other functions.

In our implementation, the Web services client is the packaging module. During the invocation process, the client creates an instance of the Web service endpoint implementing the Java interface. Each endpoint has an associated WSDL interface that defines the operations that can be performed on it. In Java Web services, these endpoints are implemented using Java proxies or stubs and invocation handles (Hansen, 2007). The client uses the proxy for communicating with

the Web service. Proxies can be created at runtime or accessed using JNDI (Java Naming Directory Interface). Generally, the client-side invocation process is the inverse of the server (Web service) invocation process. Figure 5:2 below illustrates the invocation subsystem.

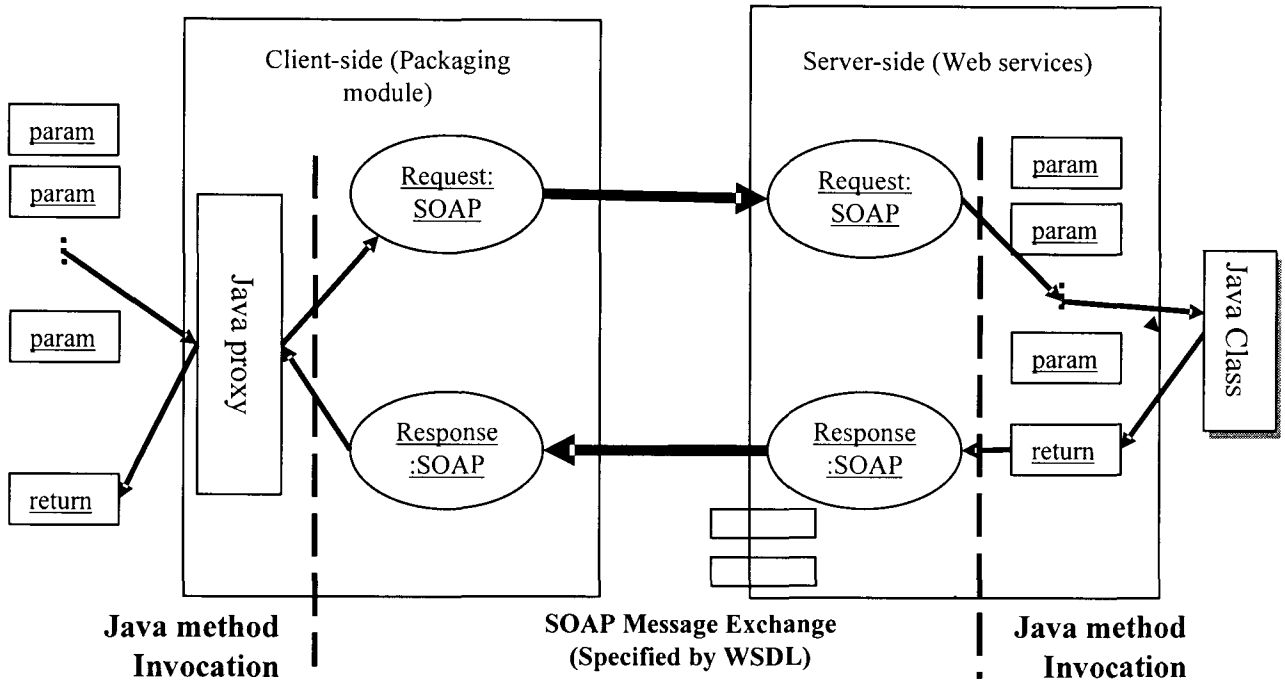


FIGURE 5:2

Invocation subsystem (adapted from Hansen 2007).

As illustrated in Figure 5:2 above, during the client Web service invocation, a method call on the proxy is translated into a SOAP request/response. On the server-side, a Web service translates the SOAP request/response into a method call on the Java implementing class.

In our implementation, the packaging module is the client and it invokes Java methods on the Web services hosted by the LMS using SOAP messages over HTTP. The calls to the Web services include requests for content, authentication and validation calls. The responses from the Web services include text documents and session IDs from Vula. Authentication calls request a session ID, after providing the administrator login and password from the packaging module. This returns an administrator session ID, which is then used to request text documents from all the publically accessible course sites. This request returns all the text files from these sites to the packaging module, which are then stored and cached locally.

The code snippet in figure 5:3 shows a Sakai login Web service Java class and an associated WSDL file snippet for its login method in figure 5:4. The Java proxy for this class in the packaging module, for example, binds the Java Interface method login to the WSDL request and response operations loginRequest and loginResponse, respectively. This proxy was created by the invocation subsystem as explained above. The packaging module invokes the WSDL operation deployed at the SOAP endpoint in the content hosting service by sending it a SOAP message, as illustrated in Figure 5:2.

```
/*Sakai login Web service class*/
public class SakaiLogin
{

      private static final Log LOG =
LogFactory.getLog(SakaiLogin.class);

      public String login(String id,String pw)
      throws AxisFault
      {

            //implementation details left out
      }

      public boolean UsageSessionService_loginDirect(String
      userId, String userEid, String ipAddress, String
      userAgent)
      {
            //implementation details left out
      }
      public boolean logout(String id) throws AxisFault,
InterruptedException
      {
            //implementation details left out

      }
}
```

FIGURE 5:3

A sample Web services class.

```
<wsdl:definitions targetNamespace="http://localhost:8090/sakai-
axis/SakaiLogin.jws">
-
      <!--
WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (05:23:37 EDT)
-->
-
      <wsdl:message name="loginRequest">
<wsdl:part name="id" type="xsd:string"/>
<wsdl:part name="pw" type="xsd:string"/>
</wsdl:message>
-
      <wsdl:message name="loginResponse">
<wsdl:part name="loginReturn" type="xsd:string"/>
</wsdl:message>
-
      <wsdl:portType name="SakaiLogin">
-
      <wsdl:operation name="login" parameterOrder="id pw">
<wsdl:input message="impl:loginRequest" name="loginRequest"/>
<wsdl:output message="impl:loginResponse"
name="loginResponse"/>
</wsdl:operation>

      <wsdl:binding name="SakaiLoginSoapBinding"
type="impl:SakaiLogin">
<wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
-
      <wsdl:operation name="login">
<wsdlsoap:operation soapAction=""/>
-
      <wsdl:input name="loginRequest">
<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input>
-
      <wsdl:output name="loginResponse">
<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8090/sakai-axis/SakaiLogin.jws"
use="encoded"/>
</wsdl:output>
</wsdl:operation>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

FIGURE 5:4

A snippet of WSDL document generated for the Web service class in Figure 5:3.

The packaging module/service is the client of the Web services hosted by our implementation. It is implemented in such a way that it periodically invokes the Web services to acquire new content. The periodic invocation is configured using an XML file and it is set as a time interval at which calls to the Web services should be made synchronously. The time interval is set up at the startup of the module and can be increased or reduced depending on the frequency with which new content is uploaded or deleted. For example, if lecture material is added once a day, then the Web service invocation interval is set to happen after every say 12 or 24 hours. This frequency helps in minimizing redundant network calls over the communication medium since many calls are likely to return the same documents if the content updates to the course sites are infrequent. Figure 5:5 shows the packaging module's UML class diagram showing the classes and their associations (relationships between classes). A detailed class diagram for the Packaging module is in Appendix F.

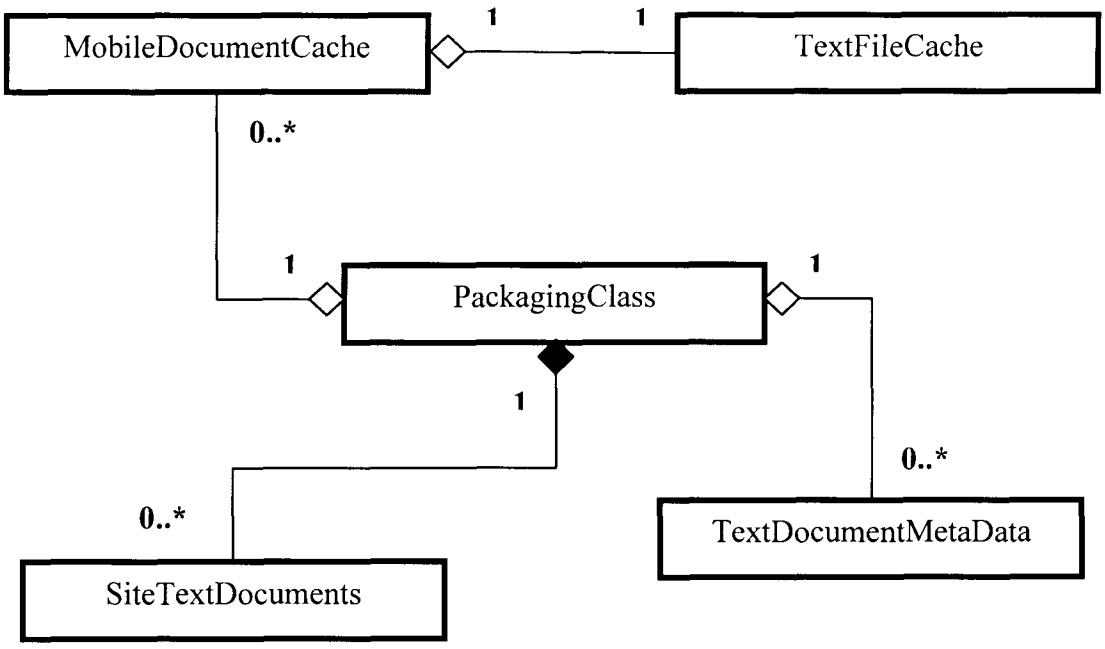


FIGURE 5:5

A class diagram for the Packaging module showing class associations.

The Distribution module as the third module in our architecture implementation comprises a distribution class DistributionEngine that implements the JSR-82 Java Bluetooth library. It, however, uses a couple of helper classes, as shown in UML class diagram in Figure 5:6 below. A Detailed class diagram is in Appendix G.

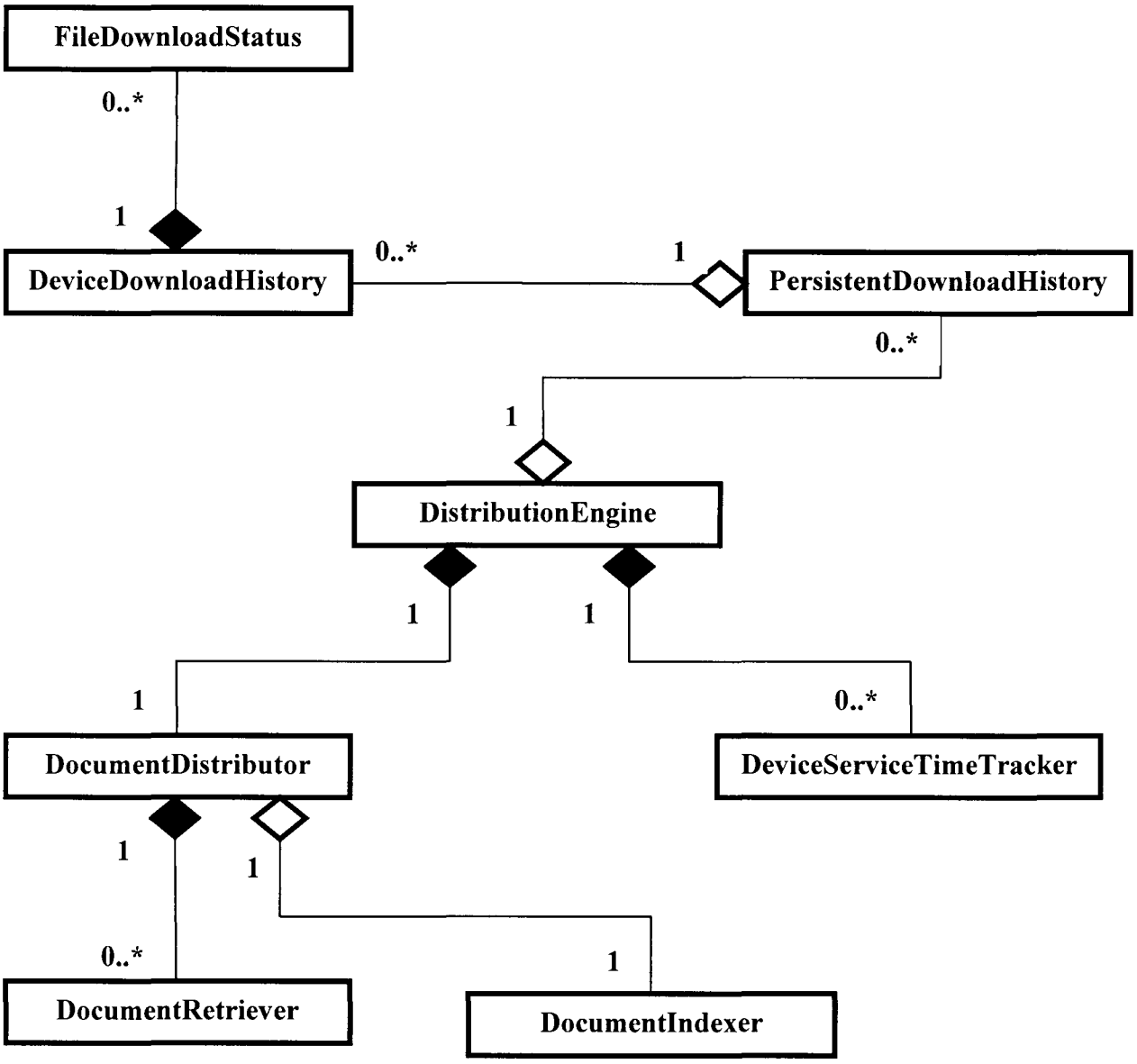


FIGURE 5:6

DistributionEngine and its helper classes.

WEB SERVICES-PACKAGING MODULE INTERACTION

In some of the current distributed architectures, services running on the same server use proprietary APIs to communicate, as per the public interfaces they expose. RPC (Remote Procedure Call) protocols may also be used by services to communicate across server boundaries. This remote service communication is accomplished through the use of local proxy stubs (as previously explained in the Web services section) that represent services on remote locations. In Figure 5:7 below, we illustrate the communication process between the packaging module and the Web services in the content hosting service

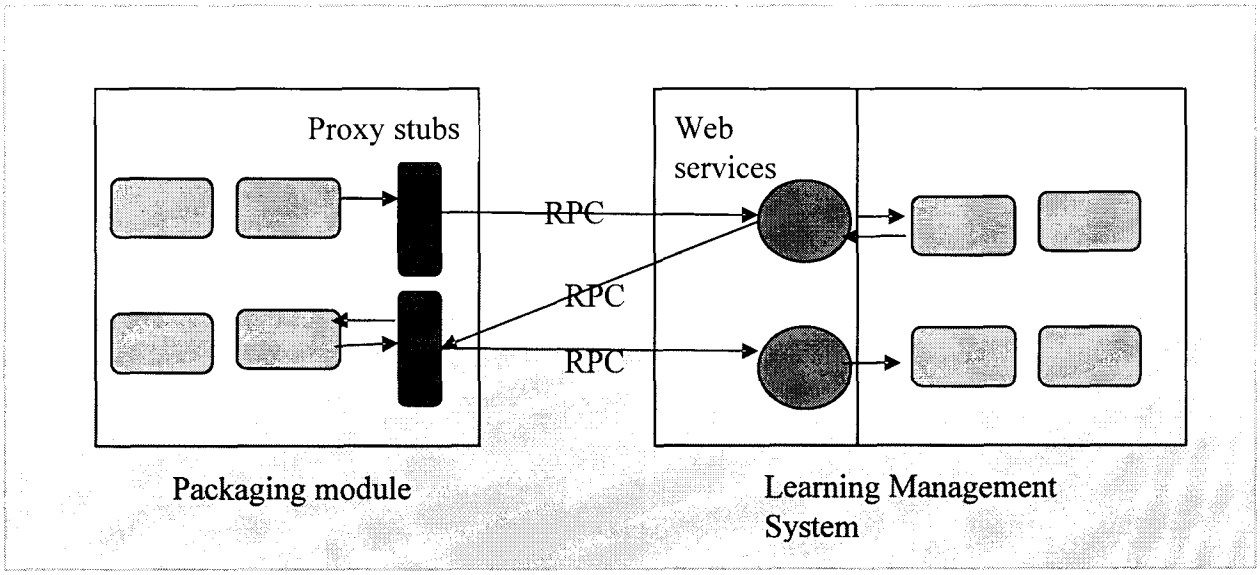


FIGURE 5:7

Packaging module-LMS communication.

In SOA, different services are designed to publicly expose a specific set of functionality (a set of operations). This functionality can originate from other services, components, legacy systems, software adapters, databases, etc. It is exposed through standardized interfaces that abstract the service implementation technology. Services therefore communicate through these standardized interfaces instead of directly with the implementing components.

Service Oriented Architectures implemented using Web services use SOAP messaging-based communication between services. This communication involves serialization, transmission, and

de-serialization of SOAP messages containing payloads (Erl, 2005b). SOAP messages are sent over communication protocols like HTTP, HTTPS, FTP, UDP, etc and are of two types: Document-style and RPC-style messages. Document-style communication involves services exchanging XML documents with one another, while RPC-style involves services invoking a set of methods remotely, i.e., through remote procedure calls (Singh & Huhns, 2005).

The majority of Service-oriented Web services rely on document-style messages for interaction. Document-style messages offer numerous advantages (Bianco et al., 2007) over RPC-style messages. Despite this, in the implementation of our architecture, we adopt the RPC-style messaging for communication between the Web services and the packaging module. This is because of their programming simplicity. This simplicity is brought about by the Web service interfaces being closer to programming language interfaces in the way they define their operations and parameters. This interface resemblance enables automatic object/class-to-WSDL translation. During our Web services deployment, the automatic translation is done by the Apache Axis Web service platform. This "hot deployment" feature offered by the platform, makes development and deployment of Web services relatively easy, since one has to only develop the Java classes and drop them in the platform for deployment at runtime as Web services.

Each Web service is designed and implemented to work alone without composition with other Web Services. In our implementation, however, the Web services use existing custom Java APIs developed for Vula to implement the required functionality. Independence of the Web services is therefore done to take advantage of the object-to-WSDL translation feature offered by the hosting Web services platform, thus making development faster. The availability of this feature and hence the "hot-deployment" creates little need to adopt the Document-style SOAP messaging which is usually harder to implement.

The interaction between the Web services and packaging module is achieved through WSDL documents that are published by the Web services platform and accessed through URLs by the packaging module. The WSDL establishes the name and location of the service as well as the data exchange requirements (Erl, 2005b), for the client to interact with.

The packaging module and the Web services communicate through RPC-style SOAP messaging as illustrated in Figure 5:7. Communication between the two services is synchronous (calls block and wait for a result to unblock) during login and content acquisition sessions initiated by the

packaging module. Each text file acquired from the LMS by the packaging module is appended with metadata that includes resource IDs, site/course name from which the content is obtained: its creation and modification dates, in addition to the text data itself. In our Web service implementation, the metadata and data are bundled in an XML document by a validating DOM parser and returned as a string at the return of each file acquisition call. The returned XML-document, is parsed by the packaging module (using a DOM parser) to extract the intelligence (metadata) and text from the self contained XML string representing the acquired text document. These returned intelligent payloads promote statelessness and autonomy of the two service modules, and alleviate processing by reducing the need for runtime caching of state (content modification and creation time) and context information (document origin) (Erl, 2005b). It also reduces the number of packaging module and Web service communication calls over the network per text file accessed in a given content acquisition session. All the XML-document serialization is done in the Web services before the service call returns. The restriction to the return values imposed by the Web service means that logical transformations (Lam & Shankararaman, 2007) must be handled in the packaging module. For example each returned string is broken down to extract metadata elements as well as text data.

PACKAGING -DISTRIBUTION MODULE INTERACTION

In our implementation, runtime communication between the packaging module and distribution module is asynchronous. It is implemented using a shared data repository as illustrated in Figure 5:8, which is a folder on the local hard drive of the machine on which we deploy the packaging and distribution module (this folder can be located on a separate machine so long as both modules know its location and can access it). The shared repository holds device registration lists and text documents that are ready for distribution to the mobile devices. When new text content is acquired from the LMS and repackaged using XML files, these XML documents are deposited in this shared folder by the packaging module. The distribution module periodically polls the shared repository and parses the XML files, creates text documents from them and deposits them in its local repository. It periodically discards the old inverted index and creates a new one that includes all the documents stored in the distribution module's local repository. The asynchronous communication model through document transfer ensures that the two service modules are

independent of each other so that one can be deployed and run independent of the other (no single point of system failure) and can be installed on separate application servers or on different operating systems or on different machines. They only need to know the location of the shared repository and not about each other. This also makes our design easily scalable to meet growing service use.

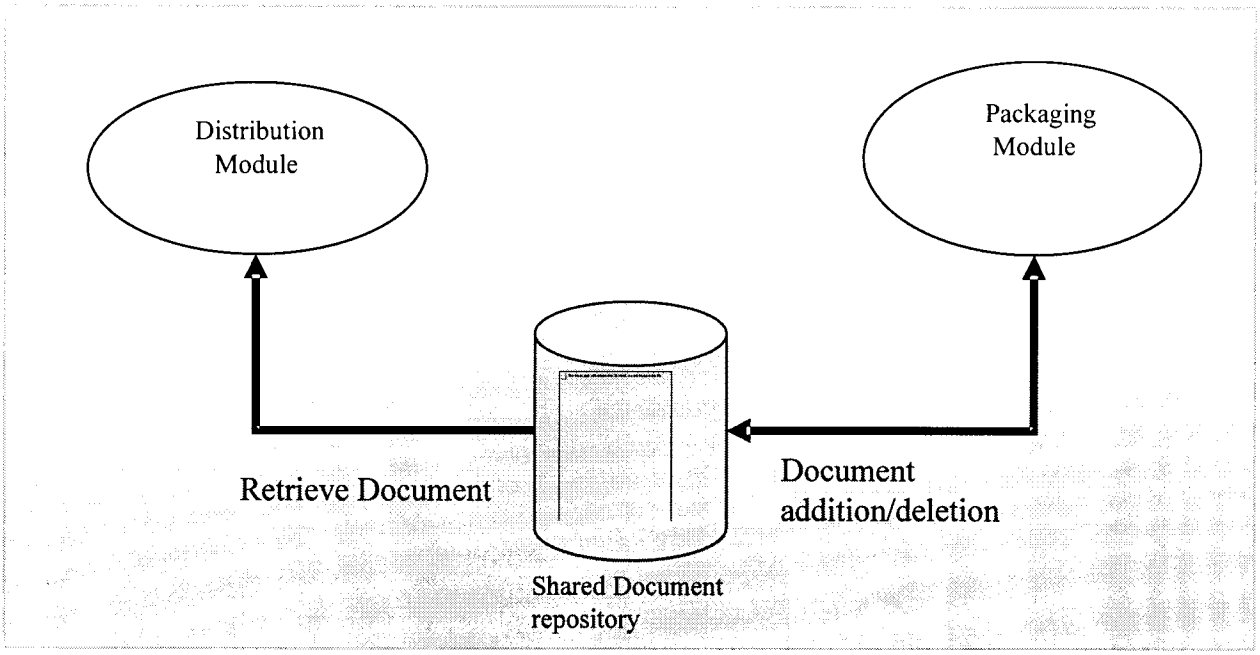


FIGURE 5:8

Distribution module-Packaging module Asynchronous communication.

STRUCTURED DOCUMENT FORMATTING

The term 'document' is a general word that takes on many definitions and thus a formal definition may fail to best fit some of its uses. For example, Andre et al. (1991) simply define document as a product of a document manipulation system. They highlight that the term also encompasses traditional products of printers, publishers, etc., such as papers, memos and notes produced in offices. We stick to their simplistic definition during the course of discussion of our educational content distribution service.

We design and implement the system to repackage and distribute text content obtained from structured documents hosted by a LMS called Vula. Andre et al. (1991) explain that a structured document is not just a stream of characters throughout which presentation information has been

scattered, but an organised structure. This structure can be logical (e.g a book consists of a preface, chapters, biblography. A chapter is formed of a title, sections, sub-sections, etc.) and physical (a chapter title is centred, 14 points, bold, etc). The structured viewpoint enables documents to be electonically processed, for example, by automatically creating an index, a table of contents, automatic numbering of sections and easier reordering. This ease of working with structured documents minimises the time taken for search and replace operations that have to be carried out in non-structured documents.

Our system manipulates structured text documents uploaded in the LMS. It leverages their logical organisation to help in processing and packaging into smaller text documents, that the distribution module pushes to Bluetooth phones and PDAs. Our system design ignores the document's physical organisation and graphical structure (The graphical structure allows the document to be printed on paper or displayed on screen). The packaging module, creates a smaller document from a large stuctured text document using either a section or a sub-section and its related text lines as logical units, in addition to the document metadata (which includes file IDs, the site from which a document was acquired, time of document creation (upload) and its modification time). The metadata is used by the system modules in tracking the document as it propagates and changes from the Learning Management System to the distribution module before it is pushed to a device.

During the initial design stage, we employed the Least Common denominator design approach (Ballard, 2007). Using this approach, we identified that most mobile phones and PDAs can render text documents with minimal quality degradation. Text files generally occupy less device memory compared to images and multimedia files. This makes them faster to download to these memory constrained devices, and they consume less bandwidth when transported over communication channels like Bluetooth and HTTP. Since our system is designed to deliver content using Bluetooth, it is practical to choose a document format that will allow as much data as possible to be pushed to the devices. This therefore explains our choice of text as the most suitable document format for this service.

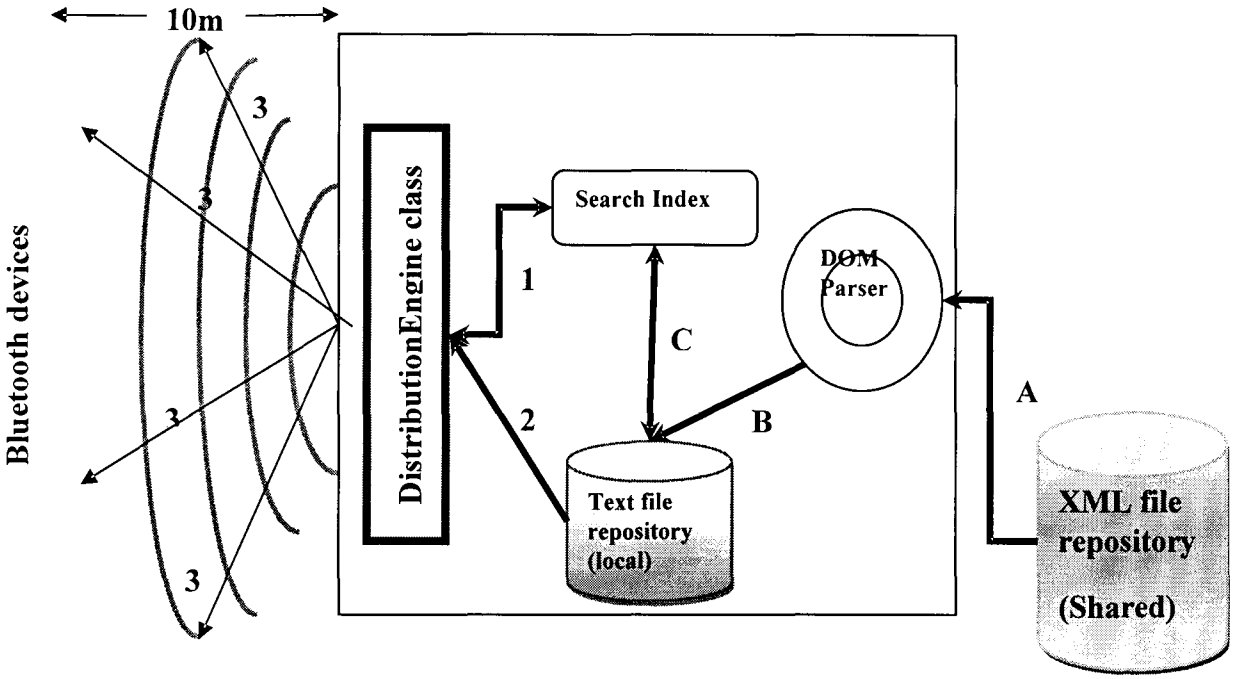
LMSs have inbuilt functionality to process, store and distribute text content in addition to other media like images, video, voice (which some may not inherently support). Since most mobile handsets have inherent text processing and rendering capability, this makes text documents the most suitable format for use in our CDA design. In the design, we ignore proprietary document

formats like PDF and Microsoft word documents. Our service requires that documents in these proprietary formats are converted to text equivalents before they are uploaded to the LMS course site resources for distribution. This, however, implies that the educational material distributed through our service maybe less rich since it may lack some informational sections like illustrations, tables and other multimedia artifacts. Text only documents nonetheless suffice for dissemination since, predominantly textual informational sections in structured documents like abstracts, brief findings, conclusions, references, introductions and other sub-sections barely contain other formats like images.

DOCUMENT PUSHING TO MOBILE DEVICES

Communication between a deployed distribution module and mobile devices is achieved using the Bluetooth communication protocol. Bluetooth is a wireless protocol integrated with many manufactured handsets and has attractive capabilities; most important of which is its low power consumption and ability to penetrate physical barriers. The Bluetooth 1.1 version most commonly deployed on devices in the market has a relatively good raw data rate of 1 Mbps and a communication range of 10 meters. In the implementation of the distribution service module, we use the Avetana Bluetooth library. This library implements the JSR-82 Bluetooth specification for Windows, Linux and Mac OS X platform. This makes the distribution module, which uses Avetana Bluetooth library, deployable on the three operating system platforms without having to alter the implementation to use Bluetooth hardware installed on them.

During runtime, the distribution module scans and discovers devices within a 10m radius and verifies which courses they are registered for. It acquires all the documents it is supposed to push to the registered device from the local document repository. For each discovered registered device, the DistributionEngine routine searches for the file transfer service on it, creates the connection to the device and pushes files to it using the Object Exchange (OBEX) profile over the Bluetooth communication protocol as illustrated in figure 5:9 below.



LEGEND:

1 Device registration authentication
2 Text document acquisition
3 Text document pushing
A XML file acquisition and parsing
B Text document storage
C Inverted Index creation and refreshment

FIGURE 5:9

Distribution module architecture showing major routines and processes.

6. DESIGN EVALUATION

THE DOCUMENT PROCESSING MODEL

We employ a document processing model illustrated in the Figure 6:1 below to reprocess the structured text documents into smaller text files. This document transformation is done by the packaging module.

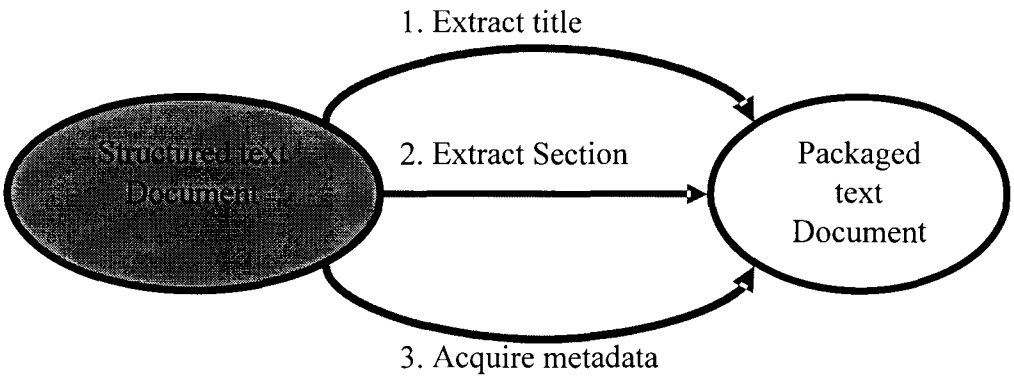


FIGURE 6:1

Document processing activities.

Our design implements a three step document processing model as illustrated above for document packaging.

The first step in document packaging involves the reading of the first line of the structured text file and this is stored as the title for the derived text document. It is assumed that this line is always the title since most structured documents begin with the title as an abstract overview of what the document is about.

In the next step, a logical section is identified and the lines that follow extracted line by line. This is appended as the body of the derived document with the title that was extracted in the first step. This section of interest is chosen and configured using a special element in the XML file that is read by the packaging module at each start-up. The section can be a heading or sub-heading of a structured text document. For example, a section with headings such as abstract, introduction, results, references or bibliography can be set as the section to be extracted and distributed from all uploaded journal documents at start-up of a deployed packaging module.

For our system to package structured documents successfully, all sections must have clearly marked headings (clear headings or sub-titles must stand alone on one line). In benchmarking the service, we use journal papers as examples structured documents (since most have clearly marked headings and sub-titles). The system is implemented to package any structured document uploaded in the LMS's public course site resource folders with clearly marked section headings.

The number of lines that make up the body of the derived document is a configurable option in the XML configuration file (Appendix H) for the packaging module. In this configuration file, an XML element defines the number of lines to extract from the chosen section.

The use of configurable section headings and predetermined number of text lines to extract, however, constrains the service adaptability during runtime. For example, at startup of the packaging module, the section and the number of lines to be packaged is discovered by reading and parsing the configuration XML file. These configurable service options, however, make the service easily customizable. The packaging module administrator can decide which section of interest and number of lines the service should distribute before starting up the service. This makes service flexible in packaging and distributing a chosen section of any structured document. In a learning environment for example, when students carry out individual experiments and there is a pre-defined format for writing their results, these results can be uploaded and distributed using the service, thus enabling exchange and sharing of educational material using mobile devices.

In our implementation, we place constraints on section headings, one of which is that each heading must be placed on a line by its own and preferably a one word description as exemplified in most journal documents. The system is designed to deal with homogeneous (Andre, Furuta, & Quint, 1991) structured text documents that consist of only alphabet and numeric characters. It cannot package mathematical formulae, line drawn objects, images and other multimedia forms for distribution.

The final stage in document packaging involves the capturing and appending of the metadata about the original file into the derived documents as they move through the modules. Metadata as explained includes the site from which the document is obtained, creation and modification time, and the document identifier in the LMS. This metadata is used in tracking documents during deletion and updating throughout the three service modules. It is also included with the documents as they are acquired from the LMS. During the document acquisition process, these

intelligent payloads help in minimizing network traffic per file between the packaging module and Web services hosted by the LMS. These payloads are self describing and once acquired, there are no extra calls made to the LMS for their subsequent processing by the other two modules.

Document creation and modification time data can have other practical applications too. For example, the service can be transformed into a news or update channel if distributable documents are also updated when the modification time for the original document differs from the one stored as metadata in the derived document. The differences between the modification times imply that the content was edited or new headlines or content were added. In our implementation, however, we do not explore this practical application since we assume that educational content is relatively less edited, but more frequently deleted or new content uploaded in the Learning Management System.

MODULE PERFORMANCE ANALYSIS

PACKAGING MODULE-LMS WEB SERVICES COMMUNICATION SCALABILITY

Although asynchronous message exchange patterns are encouraged (to maintain loose coupling) in SOA because of their service processing optimization and minimization of network communication calls, the Web services and packaging modules interact synchronously (calls block until a result is obtained by the caller or an exception is thrown by the callee). During this synchronous communication, however, intelligent payloads are returned by calls from the LMS. Each payload encapsulates the text file and its metadata. This means that, to acquire one text file from the LMS, one call is made over the network and it only returns when a file with its metadata is acquired, otherwise an exception is thrown by the Web services.

Assuming that content is updated (uploaded or deleted) in the LMS during the day but the packaging module is configured to poll for content once a day, say at midnight, then the packaging module only refreshes its local and cached copy of the text files once a day (Content updates in Vula can be done by the various site owners, users with special privileges or the system administrator). The content polling interval in the packaging module is a configurable option in the XML configuration file, a sample of which is shown in Appendix H. After the packaging module acquires content, then a cascading content update takes place throughout the

local repositories, shared content repositories and caches. This update involves deleting stale content and adding new content to these archives.

Assuming the polling interval is configured to happen with a frequency, f, during the whole day, and there are a number of files, n, currently present in the LMS after uploads and deletions, then the total number of network communication calls T_c made during that day is given by equation 6.1 below.

$$T_c = f * n \qquad 6.1$$

This is because each document is acquired with exactly one network call to the LMSS, thanks to the intelligent payload scheme. The total number of communication calls also takes into account unsuccessful calls, since the two modules communicate synchronously.

Since the packaging module polling time interval is known at startup, it remains the same for a given startup (it can only be changed by stopping and restarting the packaging module). This implies the total number of calls made by the packaging module to the LMS over a network is directly proportional to the number of documents currently present in the LMS as shown by the equation 6.2 since we assume that the polling interval remains fairly constant compared to the content updates.

$$T_c \propto n \qquad 6.2$$

The above equation shows that communication calls between the packaging module and the Web services modules hosted in the LMS, is theoretically linearly scalable.

To increase the efficiency of communication between the two modules, we need to reduce the size of payloads and optimise the speed of document parsing by the DOM parser (in this case faster parsers like StAX or SAX (Sun Microsystems, 2005)) assuming network bandwidth remains stable (which is realistically impossible, so we will have to factor in network latency within our optmisation measures). Network latencey may also negatively affect module communication efficiency when the modules are remotely deployed on different machines and thus communicate over networks like the Internet or LAN. This latency makes calls and their results take a longer time to execute. The increased delay (due to network latency), however, is inconsequential when the packaging module only polls for content from the LMS once or twice a day, which in our envisioned service deployment is the most probable case.

For more frequent packaging module polling, involving many text documents, a faster XML document parser may reduce the overall module communication time. This is because communication between the packaging module and the LMS is synchronous for a given content acquisition session. Synchronous communication means that each call to the LMS Web services blocks until content is returned, or an exception is thrown. In the LMS, each text document is parsed, recreated as an XML document using DOM and returned to the packaging module. The returned XML document is in turn parsed and recreated as a text file document by the packaging module. The XML document parsing and creation thus is a significant part of a call to the Web service whose efficiency can impact the overall module-to-module communication in scenarios involving many text documents. By using a faster XML parser overall service performance gains could be significant but may remain insignificant for a small number of documents acquired from the LMS.

For a large number of documents, increased parser speed combined with optimal network latency may yield significant performance gains for the packaging and Web service module communication efficiency. Thus to achieve optimal perfomance between the two services, network latency, XML document parsing models, and the frequency of polling by the packaging service have to be balanced.

XML OPTIMIZATION

XML is the most common data representation in SOA solutions. This is due to its flexibility, extensibility and wide adoptability, and the need for service interoperability in SOAs. It is text-based but yields payloads that are 10 to 20 times larger than its equivalent binary representation (Schmelzer, 2002). XML document processing that involve activities like parsing, validation, and transformation are CPU and memory intensive, so they may require optimization to achieve good performance within interacting services.

In our system, all three modules are implemented using the Java programming language and since the system is meant to be deployed in a single institution as opposed to cross institution deployment, we employ versions of XML documents that do not refer to a DTD (Document Type Definition) or XSD (XML schema) (Steinmetz & Wehrle, 2005). DTD and XML schema are mostly needed in XML documents that are exchanged by services that may be implemented in different languages, use complex data types, are cross-institutional and need to interpret these

documents to parse them correctly. The DTD-free documents help in reducing document parsing and transformation. By the document parsers turning off validation during document processing, there are some performance gains, as these activities are time and memory consuming. For XML document processing, our implemented modules use DOM parsers; since documents have to be accessed randomly or processed multiple times. The requirement of RPC-Style SOAP messaging to use a DOM parser also constrains our implementation to use a DOM parser instead of StAX or SAX as alternatives.

EXPERIMENT DESIGN

The system we have designed and implemented is intended to distribute educational content by packaging sections of structured text-based educational documents from a Learning Management System and pushing them to Bluetooth devices such as phones and PDAs. The service is envisaged to be deployed in learning institutions like universities in the developing world to take advantage of the high prevalence of handsets and existence of Open Source LMSs and free Bluetooth connections.

Almost all of these Bluetooth devices can consume text. To test whether the text documents created by the service can be consumed by the devices we randomly selected four different devices (mobile phones and PDAs) that included a Nokia, Samsung, Sony Ericsson, and Windows Mobile PDA from students in the University of Cape Town. We registered the devices with the distribution service to receive content from one of the sample course sites (CSC500W) on Vula. We remotely deployed the distribution module on a Bluetooth enabled desktop to act as a content distribution point. The distribution module discovered all these devices within its vicinity and successfully sent text files to them. These devices, though, display the text documents in various ways since they have different rendering capabilities and form factors. For example, the PDA has a wider screen compared to the phones and thus shows more lines of text per screen. The Sony Ericsson phone displayed the text document directly on the screen while for the Nokia and Samsung phones, the user had to navigate to a given folder to view the received documents. These different text rendering capabilities and form factors on the various devices are beyond the scope of our research, however.

In benchmarking the system, we determine distribution module response time during peak (stress load) and off-peak (average load) service usage contexts. In benchmarking service response time for the distribution module, we intend to determine what the most suitable deployment environment will be for the content distribution points and how many distribution modules may be deployed to suitably serve a given number of users. In our implementation, the time it takes a device to receive all the documents it has been subscribed to, includes service response time and file download time. Service response time includes device discovery time, and connection establishment time. All this time must be spent by the distribution module before a user knows whether they have content or not (a user can only know this when they get the first response from the distribution module after they have Bluetooth-activated their devices within a 10m radius of the distribution point). After the distribution module establishes a connection with a registered device, it pushes the files the user is meant to get one by one. This actual pushing of documents to a device is what we term as the content download time.

In evaluating the efficiency and scalability of the distribution module, we design an experiment to investigate the effect of text document number/size on the content download time. In this experiment, we register three PDAs (using their Bluetooth IDs) for the service to receive content from one course site on Vula. In the distribution module, we automatically log the time just before (T_b) the call to push documents to a device is made in the distribution module, and the time after (T_a) it completes (meaning all the registered-for documents have been sent to the device). In our implementation, the content pushing routine only returns after all the documents a device is registered to download have been pushed to that device. The difference between these two times (T_l) is the time it takes to send a number of files (of a given size) to a device, which is mathematically formulated in equation 6.3 below.

$$T_l = T_A - T_B \qquad 6.3$$

In the experiment we use text documents that average 750B in size and consist of 12 lines of text in the body. These files contain the course site from which the document is obtained, the title of the original document from which it is extracted and a body that consists of ten lines of text from the section of interest. For example, Appendix B, shows a text document containing 12 lines of text extracted from the abstract section of a paper from course site CSC500W.

In the experiment we upload a number of structured text document files to that course site and wait for the content distribution module to push them to three PDAs (after repackaging by the packaging module). For example, in the first run of the experiment we send one document file to each of the three PDAs, the next run we send 2, the next 3, 4, and so on. For each run we capture three content download times using equation 6.3 on each. We calculate the average of the three download times for each run and use this as the content download time for that pushed number of documents. We calculate the average of the three content download times per run to mitigate error that may occur during connection establishment and document transfer to the devices. In our experiment, we carried out 10 such runs and plot the graph in figure 6:2 to show the effect of document number/size on the content download time. We append the obtained raw time data used to draw the graph in Appendix I.

Use of similar devices, in our experiment PDAs, ensures that there is uniform content downloading procedure by the users in the experiment. All these PDAs accept a single Bluetooth connection per group of documents pushed to them, and the documents are downloaded to a location in the device. This download procedure may differ when using a Bluetooth-enabled Sony Ericsson phone, which requires a user to accept or deny each file being pushed to it, while in some Nokia phones files may be pushed without prompting the user to accept or deny Bluetooth connections. Using similar devices ensures that there is uniformity in the content download time trends due to the use of a consistent download procedure in the devices. Since the essence of this experiment is to investigate how the number/size of text documents pushed to a device affects the content download time, our results using only PDAs sufficiently show this effect (though it could vary in magnitude depending on the device content downloading procedures).

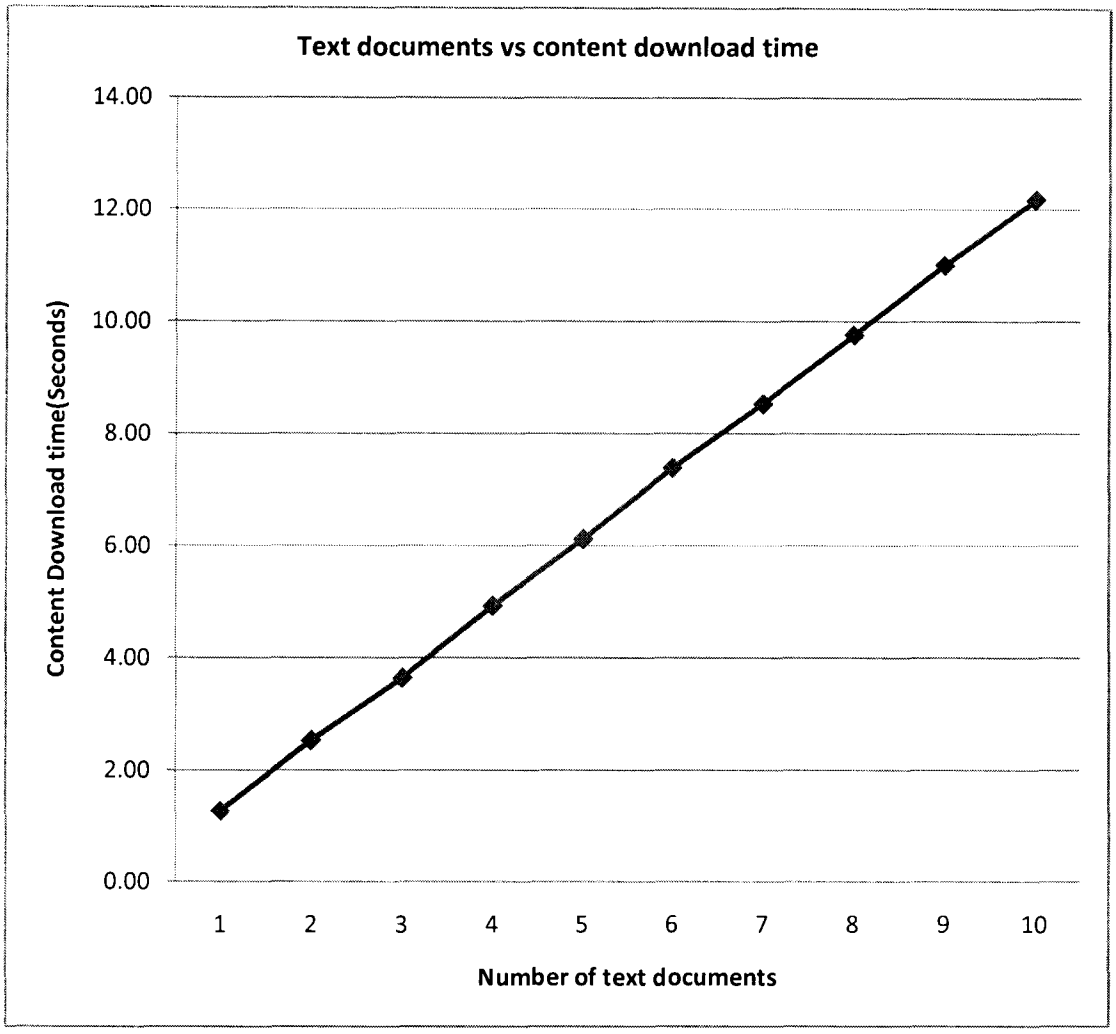


FIGURE 6:2

A graph showing the effect text file number on content download time.

Results from the above experiment show that in the current implementation of the distribution service, the content download time by a device linearly increases as the number/size of text documents pushed to a device increases. Using PDAs, the results show that each text document of roughly 750 Bytes will take 1.22 seconds to push to a device. This implies that if many users are each downloading many files at the same time, the overall response of the service may degrade, as each user will have to wait a time that is a multiple of the number of documents he/she has to download. The service may, however, sufficiently respond to users when they are downloading a reasonable number of files.

SERVICE RESPONSE TIME

We define service response time as the time that a device takes from having its Bluetooth activated to the time of successfully establishing a connection with the distribution module (just before the first document is pushed to it by the distribution module). This time amounts to how long a user will have to wait before they get a response and possibly files from the distribution service. We measure the service response time by defining scenarios and simulating two service usage experiments in these scenarios (Jones & Marsden, 2006). These scenarios are peak (stress load) and off-peak (average load) service usage contexts. In both experiments we obtain the average service response time and compare the results. We then conclude and give recommendations based on the results of how best to deploy the distribution modules. Shorter response time will mean that the service is more responsive to users in that scenario, thus making deployment in that environment more suitable than the other. Almost equal response times in both situations means that the service can be deployed anywhere on the university campus. Longer service response will mean that if the service is to be deployed in heavy usage contexts, then many distribution service modules will have to be deployed to split the user load (which is an expensive alternative).

OFF-PEAK RESPONSE TIME EXPERIMENT

During the off-peak service use scenario, we envisage that the user is in a stationary position for some time, say, seated in a library, having lunch in the cafeteria, chatting to a friend while standing, and waiting for a bus, and so on. The user is doing any of these activities within a 10m radius of the distribution module. The user therefore can do other things while their Bluetooth is activated to receive content from the content distribution point.

To simulate this environment, we imagine users sitting in the library as a case in point. In our experiment, we use a Bluetooth enabled computer running the distribution module and packaging module of the educational content distribution service architecture. This computer communicates with Vula (the LMS), which is hosted by another computer on the same local area network. We use nine Windows PDAs and three users. These devices are registered to download content from a given course site (CSC500W) on the Learning Management System. Each device

is tagged with a name and a number to easily identify it during the experiment. These device numbers are consecutive from 1 to 9, marking the nine devices.

The three users are given three devices each; user one is given devices numbered 1 to 3, user two devices 4 to 6, and user three devices 7 to 9. All the devices use a stylus and the users are taught how to activate and de-activate the Bluetooth connection. Each device is configured to be discoverable by other devices and to allow all other devices to connect to it. Each device authorizes an incoming Bluetooth connection once by prompting the user to allow or deny a connection from a given device.

To model a library environment, we set up tables for each user in a room. The tables are distributed in the room within a 10 meter radius of the distribution service. A beeping routine is deployed on the distribution module which prompts the users to activate their devices. The routine makes a beep after every 20 seconds.

During the experiment, the distribution module is started and then the beeping routine is started too. The first beeping sound is ignored and is meant to alert the users to get ready to activate the Bluetooth on their devices. For each of the remaining 9 beeps, a device with that number is activated. For example, on the first of the nine consecutive beeps, device 1 is activated, second beep device 2, third beep device 3, till device 9. The beeping routine logs the time (T_a) in milliseconds each beep is made.

Each activated device then waits to be discovered, verified, connected to and sent the files that it registered for by the distribution module. For each device, the time (T_b) just before the first document is sent and the device name are logged to a file by the distribution module. We choose to log the time just before the user is sent the first file, because, as earlier shown by an experiment, actual content download time increases with the number of files a user has to download. Since most users may be registered for different content, the times users will take to download all their content will vary.

After receiving the documents, the user deactivates the Bluetooth connection to free that Bluetooth channel. This is done until all the nine devices have received documents from the service and this is the end of one experiment run. The predicted user behavior is that after users receive their files, they will most likely walk away (out of the distribution module's 10m active range) from the distribution point or switch off their Bluetooth.

The service response time for each device per run (T_i) is calculated using equation 6.4 below.

$$T_i = T_a - T_b \text{ where } i \text{ is the run number} \qquad 6.4$$

For this experiment we performed ten runs thus obtaining (9*10) response times. The raw response time data obtained from this experiment are shown in Appendix D. The graph in figure 6:3 below shows the histogram plotted from the data.

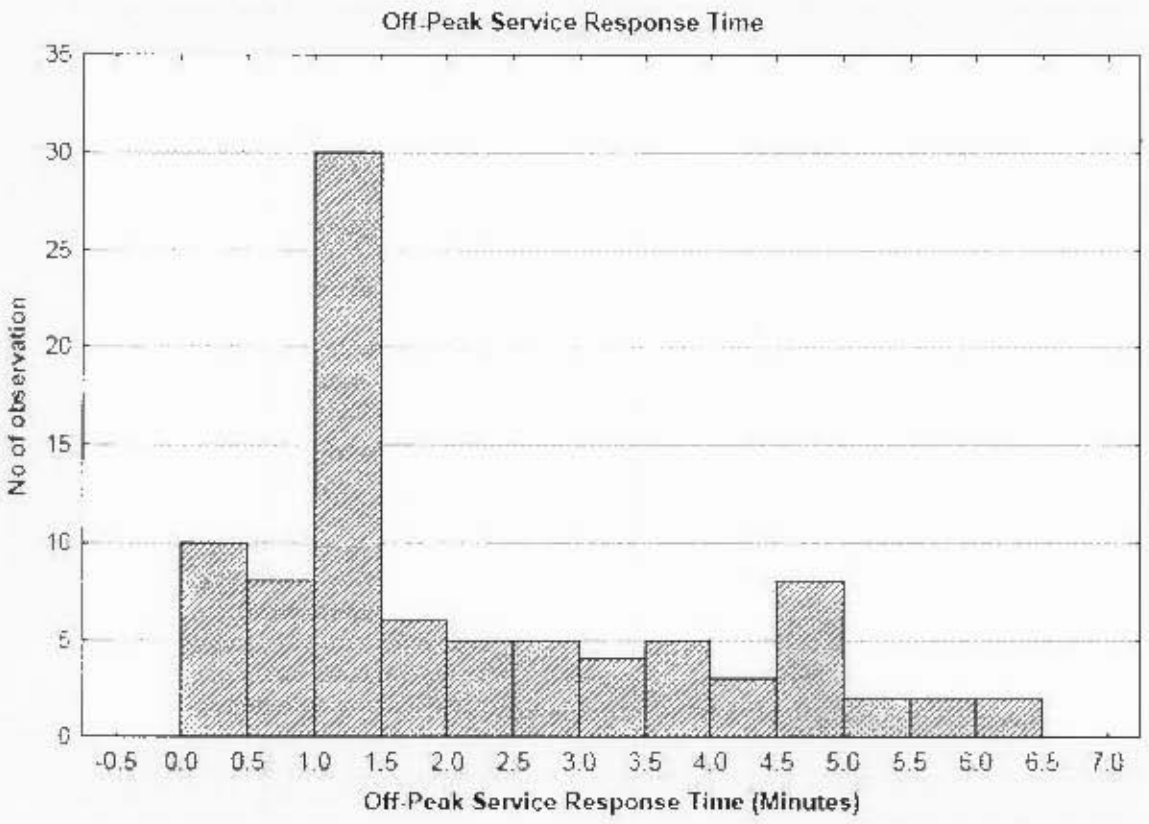


FIGURE 6:3

A graph of off-peak service response time.

The mean response time obtained for the 90 response times is 2.20 minutes during the off-peak usage (average use) experiment. The values obtained have a standard deviation of 1.633872. This service response time means that a user has to wait for at least 2.20 minutes to know whether they have content to download or not and thereafter begin getting the actual documents if they have any. This may be a reasonable waiting time since the user in this context is not very concerned about delay and is most likely to spare a few minutes to download the content as they do other things.

In this usage scenario, we envisage a busy environment with many users who are relatively mobile and more time conscious than off-peak users. Many registered devices will be activated within the 10m range of the distribution service in a short period of time. This will imply that the distribution module will be subjected to handling many Bluetooth connections in a short period of time. Since each Bluetooth connection can handle a maximum of 8 devices concurrently, in the worst case scenario of eight Bluetooth connections being concurrently served, the service may be unable to handle extra connections until one or more of the connected devices has been served or goes out of range. Taking students walking out in a corridor to the next lecture venue, or students that have just ended a lecture and are waiting to go to the next one, as cases in point, we simulate the peak usage context.

In this experiment, we register nine marked PDAs as in the previous experiment (for measuring service response time in off-peak periods) and still use nine devices but with nine users this time. The users are taught the same Bluetooth operation procedures as in the above experiment. Users are, however, mobile within the room and activate their Bluetooth connections all at once when they hear the beeping sound from the beeping routine. The beeping routine is configured to make two beeps for each run of the experiment. The first beep is the alert sound and at the second beep, all users activate their Bluetooth connections. The time (T_a) at which the second beeping sound is made is recorded by the beeping routine and the distribution module records the different times (T_b) at which each of the nine devices are sent files as in the above experiment. Each user deactivates his/her Bluetooth after downloading the files. Each experiment run is ended when all the users have got content and their Bluetooth connections are deactivated. Response time is calculated using equation 6.4 above.

To reduce error in the measurements, we use repeated measure technique (Howell, 1989), on the nine PDAs and carry out 15 experiment runs from which we obtain data. We discard the first few trial runs since users are assumed to be learning the device operation procedures and thus the results may be error prone. Users are allowed to practice operating the PDAs, since in reality, each user is assumed to be familiar with the Bluetooth operation of their devices. We obtain the mean service response time in minutes for the last 10 runs which gives us (10*9) response times. The results obtained are tabulated in Appendix E. We plot a histogram for the response times from this experiment as shown in figure 6:4 below.

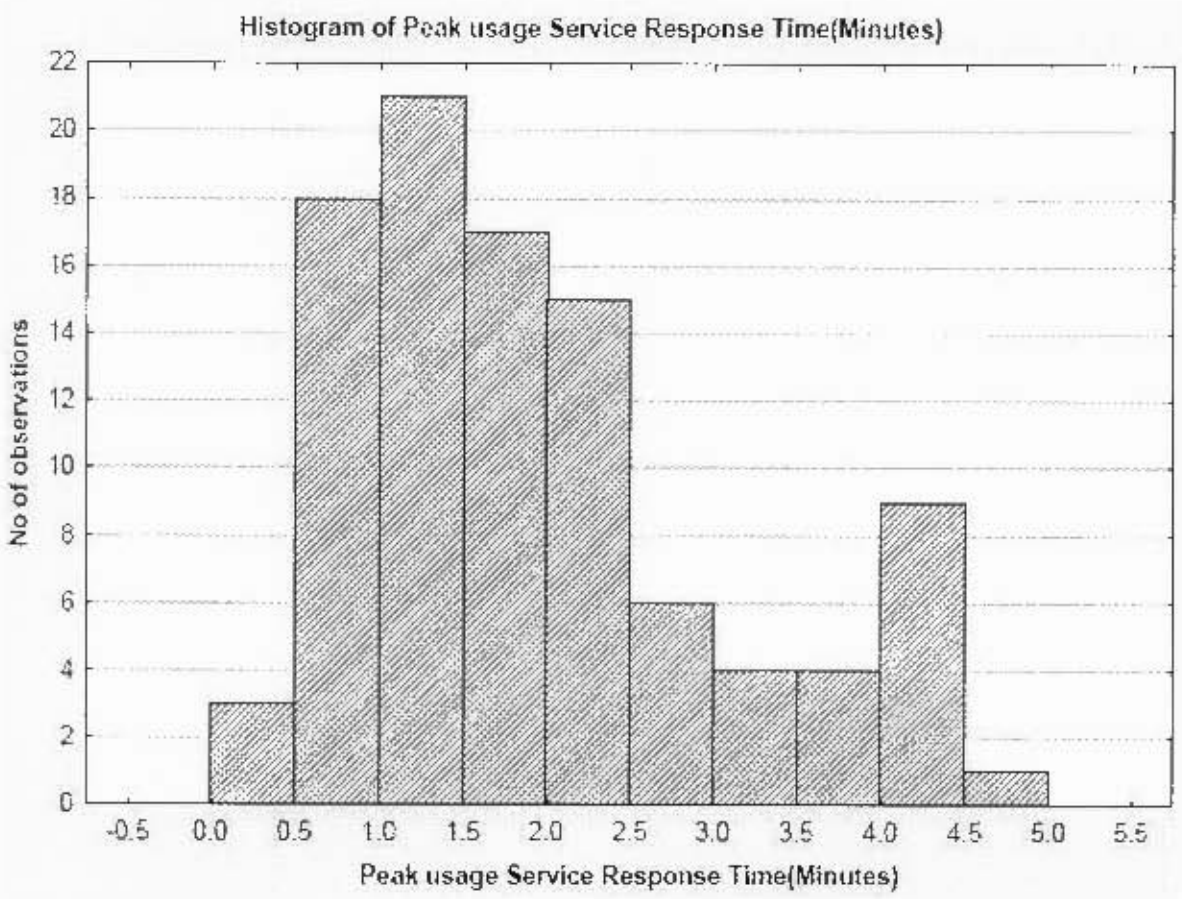


FIGURE 6:4

Peak (stress) load usage service response times.

The results obtained from this experiment with 90 response times show that average service response time during a simulated peak usage scenario for the service is 1.94 minutes. This implies that a user has to wait for roughly 2 minutes to get a response from the service i.e. to know whether they have content to download or not during busy service usage times. Practically, if a user hurriedly passes near a deployed distribution module, they might not know whether they have content unless they cover the service discovery range of 10 meters over at least 2 minutes.

The results from the service response time experiments show that the service responds to most of the users between half a minute and three minutes. During this time period 58% of the devices were discovered during off-peak usage with 79% of the devices in peak usage. The mean service response time during peak usage is 2.2 minutes while in stress load situation its 1.9 minutes. The standard deviations between the response times are 1.633 for off-peak usage while it is 1.125 for peak usage. The lower average service response time during the stress load situation may be caused because devices are switched on almost at the same time (when the beep sounds). This means that the Bluetooth service frequency-hops and discovers devices in groups for example in groups of 4, 5, 3, 7, etc in a given run. In our implementation, once devices are discovered, they are added into a Java hashset collection one after the other in the order of discovery. After the device discovery operation, services that are offered by the devices are searched for in the same order. For each device whose service is searched for, a Bluetooth connection is established and all its available content is sent to it. For a given discovery run, all the devices are discovered, and then the distribution module pushes their content to them. This implementation means that when devices are discovered in groups, they will receive content one after the other, depending on the order in which they are discovered. The group device discovery phenomenon has an overall effect of bringing down the average service response time and this explains the lower average service response time for peak usage. Raw service response time data in Appendix E shows the group discovery trends for the distribution module in the peak-usage context.

In off-peak usage, however, each device was switched on after an interval of 20 seconds (signified with a beep sound) as noted in the experiment procedure. This interval may have caused only one or smaller groups of devices to be discovered during each discovery-service search-Bluetooth connection cycle. The smaller groups mean fewer devices are sent content per cycle since all the devices discovered in a given cycle have to be sent files before another cycle begins. This means that more cycles are made before each experiment run is ended in off-peak than in peak usages. As observed, each device discovery sub-state may take at least 10.24 seconds, meaning cycles will have a cumulative effect on the overall time spent to discover the devices if the devices are discovered in smaller groups (as this is the trend in this experiment). This may explain the longer average service response time during off-peak usage. Raw response time data for this experiment is attached in Appendix D.

7. DISCUSSION AND CONCLUSION

In this study, we have proposed, prototyped and evaluated an educational content distribution service to Bluetooth enabled mobile devices (targeting mainly mobile phones and PDAs). Our service extends an existing Open Source Learning Management System called Sakai (which is branded Vula, for the University of Cape Town) by leveraging its existing Web service extensibility features. Sakai is one of the most popular Open Source LMS and offers similar functions and tools to other LMSs, one of which being the ability to be extended via Web services and its related XML specifications. This extensibility creates an avenue through which services and tools offered by the LMS can be extended to interoperate with other systems. Extending the content distribution service to mobile platforms is one example of extending the currently available Web-based content distribution channel offered by Sakai. In our service we have targeted the availability of Bluetooth enabled handsets as these offer a free communication channel that will make the service affordable (without introducing any new costs) for most students in universities, especially in the developing world.

In our design we have looked at providing an affordable service that can be integrated into the daily learning routines of students. We have leveraged existing infrastructure, specifically Bluetooth enabled handsets, within student communities in these universities and the already existing LMSs by integrating the two disparate technologies to create another channel of educational content distribution, in addition to the Web-based and classroom-based ones. Our solution has targeted the Open Source LMS to keep it cheap, since no extra licensing fees will be needed by the universities to add this service to their existing installations.

The service design is based on Service-Oriented Architecture paradigm, which is a vendor neutral architectural model. This paradigm has many characteristics that make it suitable to ground our service design, most notably service/module reusability, loose-coupling, service composition, emphasis on extensibility, and is based on open standards and communication frameworks like XML and Web services. Based on this model, we implemented a functional prototype to package and distribute text documents, created from uploaded structured text documents within Sakai (Vula), to Bluetooth enabled mobile devices.

The results from our experiments show that our system can be deployed in busy environments like corridors. The user though, will have to be made aware that they need to wait for a few

minutes to get content from the distribution nodes (a notice must be used to show the minimum number of minutes before they should begin expecting content from the system after they have activated their Bluetooth). For large classes, there should be many distribution nodes so that the registration lists can be split among the existing nodes so that users do not have to wait for very long periods (which could be the case when one node is pushing content to a long list of active devices). These nodes can be close to each other and users need only be informed of their presence.

Deployment in light usage areas like libraries and student bus stops (where a user has time to wait while doing other things) is also possible with fewer distribution nodes compared to busy scenario usage. Here, a node can have a device registration list of all the students in a course. Since the users presumably have more time to wait, this deployment scenario will need fewer nodes than the busy usage scenario. In the light usage environments, a notice to show the presence of a distribution node and, say, the courses whose content is pushed from that node can be used. The notice can help to inform users whether they are in the vicinity of the right node or not. A distribution node can be any PC or laptop that is Bluetooth enabled and is running an instance of the distribution module.

ARCHITECTURE OVERVIEW

We implemented our design using the Java programming language (however, could have implemented it using any programming language due to SOA model vendor neutrality). In our design we created three modules: the Web service module, packaging module, and distribution module. The Web service module offers an abstraction of the implementation logic of the LMS by offering a standard interface in the form of WSDL access points that can be used by any other application irrespective of their implementation technologies. This module is used by the packaging module to login to Vula to acquire text documents that have been uploaded in the different course sites. The Web service module is hosted in the LMS.

The packaging module is the middle module of our n-tier SOA modelled architecture. It periodically polls for and acquires the uploaded structured text documents from the LMS (Vula). It locally caches these documents which it then repackages into smaller text documents using XML as an intermediate format. The smaller documents are packaged according to a pre-configured module start-up option that determines a section of interest from the whole document

that will be packaged during the subsequent running of the module. Communication between the packaging module and the Web service module is synchronous and uses the RPC-style SOAP messaging.

The third and final module of our service is the distribution module. This module parses and creates the final text documents from the XML documents created by the packaging module and deposited in a shared repository. It communicates with the packaging module asynchronously through the shared repository by reading documents that have been deposited by the packaging module. It has no knowledge of the existence of the packaging module but knows the location of the shared repository. This design decision helps in decoupling the packaging and distribution modules, thus helping the easy scalability of the services. The distribution module also scans for Bluetooth enabled devices within a 10 meter radius of its deployment, authenticates their course registration, and pushes text documents to them (documents that have been acquired from the course sites they registered for). The location of the shared repository is a module-start-up configurable option, in addition to the interval which the distribution module takes to see whether there is any new content in the shared repository, and the device scanning time intervals. The module pushes the text documents to the devices via Bluetooth connections.

The loose coupling of the service modules implies that the three service modules may be deployed on separate hardware and different operating systems, thus making our service fulfill one of the key SOA model principles of vendor neutrality. This also creates an easily extensible architecture, since many packaging modules can be installed to acquire content from an LMS and many distribution modules can be installed to distribute content from each installed packaging module making our design very extensible.

We have used XML as an open standard for module communication and as a text content data transportation format. This implies that our modules have intrinsic interoperability which makes each module a potential integration point. This characteristic can significantly reduce the cost of future cross-application integrations requirements and thus future proofs our design. For example, the small XML documents created by the packaging service can be parsed by a new application that may distribute it using WiFi connections, or via an SMS medium.

IMPLICATION OF THE FINDINGS

In the first experiment, that investigates the effect of file number/size on the content download time, we discovered that, the greater the number of files to download by each user, the longer that user will take to have all his/her content pushed to them. This means that distribution modules should be deployed in such a way that users can download their existing content as fast as possible to avoid accumulation of content which may make content download queues longer (thus increasing content download time for the subsequent users, when in fact one user is downloading a lot of files). To mitigate this linear growth in download time (of users downloading too many files at a time), a duration for which content will be allowed to stay in the distribution module could be set. This should lead to documents only being held in the content distribution module up to a certain time, after which it is assumed that all the subscribed users have got it and it is discarded or uploaded to another location. This will ensure given these assumptions that all users averagely have the same number of documents from a given course site which they have registered for.

SERVICE LIMITATIONS

Our current service implementation has limitations which may manifest themselves during deployment and /or scalability. The SOA model advocates loose coupling between services which when implemented offers advantages such as easily replaceable services with minimal disruptions to the user, addition and removal of services as demand varies, ability to locate services on any server, and fault tolerance. However, its broad-based realization is not easily achievable. In our design we achieve and leverage some of its advantages through the use of asynchronous communication between the distribution module and packaging module.

However, the packaging module and Web services are not loosely coupled due to the use of RPC-style SOAP messaging. In the implementations of the two modules, the packaging module reads the Web service description (WSDL) and creates a proxy or stub with methods that correspond to the service interfaces. The tight coupling between the services means that the packaging module must know the location of the Web services module at compile time which means there is no need for a service registry. This design decision inherently limits the back-end

scalability of our design in that only one LMS can be used as an educational content database during a particular compilation and runtime cycle of the two services. This makes our design less flexible since the location of the Web services (hosted by the LMS) cannot be easily changed without having to recompile the packaging module. It also makes the two modules tightly-coupled and less fault-tolerant because, if the Web services are unavailable, then the packaging module cannot dynamically query a service registry for an alternative LMS that can provide the same functionality. This design decision, however, does not take away the scalability, flexibility, replaceability, and fault tolerance of the service from the packaging module side. For example, many packaging modules can still be deployed to access content from an installed LMS, a packaging module can go down without affecting the LMS, and each module can be deployed on a different server with a different operating system.

The current implementation of the service only repackages and distributes educational text content through the use of a configurable XML option at the start-up of the packaging module. This limits our service in the area of its practical deployment, for example it cannot repackage images and other multimedia content for distribution to Bluetooth devices even though these devices have the capacity to render and effectively consume these content formats. The Bluetooth wireless communication also does not restrict the format of content that can be transmitted, though our design constrains it through the packaging of only text documents under the assumption that educational content is predominantly text.

The service reliance on only Bluetooth for the distribution of content naturally brings with it the technology's current limitations. Bluetooth as a technology is limited in the number of concurrent connections that can be made by devices, the data transfer rates per connection, and servicing range. For example, the prevalent Bluetooth specification on handsets supports a maximum of 8 concurrent device connections, has a maximum transfer rate of 1MB/s, and has a maximum range of 10 meters. These limit the size of documents that can be pushed to devices, the number of users that can be served concurrently and the maximum range and dictate that distribution modules be deployed in the most suitable locations in addition to deciding the number of users that can be effectively served.

SOA as an architectural style offers notable advantages to service design such as service scalability, aggregation, extensibility, composability and abstraction of implementation details. However, its realization through technologies such as Web services brings to the fore some of its

weaknesses as Srinivasan & Treadwell highlight, which include immature technology, lingering confusion in some areas of standards, and the significant performance overhead incurred by services during serializing, de-serializing and parsing of SOAP messages and XML documents during message exchange (Srinivasan & Treadwell, 2005).

Dynamic service discovery by modules during runtime, as strongly advocated for by the SOA paradigm, may also create generate unnecessary network traffic in cases where the services being looked for reside within the same organisation. Since the locations of these services are known, there is no need for services to consult a service registry to ascertain their locations (generating unwarranted network traffic). They can as easily be accessed directly by the other requesting services. The direct Web services-access approach is the one we use in the implementation of the packaging module and Web service module. We envisage the whole service infrastructure to be deployed within a university, where the location of the LMS is known and can be accessed by all the deployed packaging modules within the campus, which creates less need for dynamic service discovery.

WHAT IS NEW IN THIS SERVICE?

In most of the existing educational content distribution systems in learning institutions, the content distribution medium is predominantly Web-based. In these systems, content varies from text to multimedia. Some of these systems, in addition, can also distribute electronic library content on the Web-based portals. The registered students access the educational content using mainly Web-based interfaces, although new schemes like RSS are being used for frequently changing material as exemplified by the CTools service. The tools that aim to use the other portals to further push educational content into new channels like mobile devices predominantly exploit SMS service which is not free.

Bluetooth as a free content distribution medium is still less exploited in educational institutions but is popularly exploited in the commercial domain for marketing and advertising in systems like BlueCast, Jellingspot, BroadTooth. These commercial systems distribute content ranging from multimedia to text. Due to their use in marketing and advertising which is location specific, most of them have content uploaded directly on the distribution modules (at times called point servers). Some of them target specific operating systems for their deployment, or target specific

brands of Bluetooth handsets. Some have clients specifically developed to be installed on the mobile devices before they can consume the distributed content.

In our system, we design an education content distribution service that uses an existing Open Source LMS that is extensible through Web services. It distributes the content to Bluetooth enabled mobile phones and PDAs. This implies that it opens another channel of content distribution other than the available Web-based and sometimes SMS medium which have their limitations. The use of Bluetooth technology (which is becoming a standard on most shipped handsets) for content distribution means that the service is free to students who have Bluetooth enabled phones and thus makes it a feasible alternative in a developing world university context.

Our designed service requires no customized client software to be developed for the mobile handsets (unlike some commercial systems) and only distributes text content. These design attributes mean that the distributed text documents can be consumed by a wide array of handsets that must only have Bluetooth capability.

The system modules are implemented in Java, which gives the service an inherent capability that it can be easily deployed on any Operating System. This distinguishes it from many commercial content distribution systems that target specific operating systems and sometimes specific brands of handsets. The solution we have developed uses the existing infrastructure (Open Source LMS) like some of the related education content distribution services and tools but opens the Bluetooth channel to the academic environment as a viable alternative to the Web-based and expensive SMS channels that currently exist. It is thus well suited for developing world universities where Open Source LMSs are popular and there is a high proliferation of mobile phones amongst students.

RECOMMENDATIONS AND FUTURE WORK

Since most popular LMSs support open standards like HTTP and Web services and its specifications like SOAP, and optionally UDDI, we recommend a future redesign of the packaging module that will leverage this (use a service repository like UDDI) support, which will decouple it from the Web services module. This could be achieved through the adoption of the Document-style SOAP messaging between the packaging module and Web services module to replace our current RPC-style messaging. This could allow for dynamic discovery of the Web services descriptions through the use of registry technologies like UDDI. The decoupling of the

two modules would alleviate the problem of having to recompile the packaging module each time the location of the LMS is changed, and could bring fault tolerance to the two modules by allowing for addition of alternative or back-up LMSs to the service (since they can be added to the system dynamically without having to recompile all the installed packaging modules).

Although our service is constrained to distribute only text documents to Bluetooth enabled phones and PDAs, the SOA paradigm on which we based our design is flexible enough to mitigate this constraint if there is future need for distribution of multimedia educational content. As a recommendation for future work in this area, we foresee the creation of an extension to the packaging module that should repackage multimedia educational content acquired from the Learning Management System and put it in the shared repository for the distribution module to push to registered devices.

The choice of Bluetooth technology as the service channel used for content distribution inherently constrains our distribution service. In the meantime this could suffice, as most mobile devices are currently Bluetooth enabled, and have feature and memory limitations. However, as communication technologies become cheaper, some of these limits will be broken and more powerful wireless communications like WiFi could become standards on shipped mobile devices. By basing our design on the SOA paradigm, these future technologies could be leveraged through addition of another distribution module to our service that could be used to distribute the content through these new channels without having to scrap or re-implement the service as a whole.

CONCLUSION

The design of our educational content distribution service based on a Service-Oriented Architecture paradigm, future proofs our design so that it remains usable in an education domain in which Learning Management Systems will continue to be used. The leveraging of existing infrastructure coupled with the Open Source nature of some of the popular Learning Management Systems in our design, brings to the fore a solution that is cheap and easy to use and replicate in developing world universities.

REFERENCES

Aberdour, M. (2007). *Open Source Learning Management Systems.* Brighton: Epic.

Alreck, P. A., & Settle, R. B. (1995). *The Survey Research Handbook* (2nd edition ed.). Chicago: IL:Irwin.

Andre, J., Furuta, R., & Quint, V. (1991). *Structured Documents.* Cambridge: Cambridge Univesity Press.

Ballard, B. (2007). *Designing the Mobile User Experience.* USA: Wiley.

Barry, K. D. (2003). *Web Services and Service-Oriented Architectures.* San Francisco: Morgan Kaufmann Publishers.

Bianco, P., Kotermanski, R., & Merson, P. (2007). *Evaluating a Service-Oriented Architecture.* Carnegie Mellon University. Software Engineering Institute.

Bluetooth SIG. (2009). *Basics*. Retrieved September 9, 2009, from http://www.bluetooth.com/Bluetooth/Technology/Basics.htm#5

Cheung, S. (2004). Fun and games with mobile phones: SMS messaging in microeconomics experiments. In C. M.-D. R. Atkinson (Ed.), *Proceedings of the 21st ASCILITE Conference* (pp. 180-183). Perth: In R. Atkinson, C. McBeath, D. Jonas-Dwyer & R. Phillips (Eds).

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., et al. (2002). *Documenting Software Architectures: Views and Beyond.* Boston, MA, USA: Addison-Wesley.

Dueber, B., & Hollar, S. (2006). *Providing Library Reserves to Sakai using RSS.* University of Michigan.

Erl, T. (2005a). Introducing SOA. In *Service-Oriented Architecture; Concepts, Technology, and Design* (pp. 37-51). Prentice Hall.

Erl, T. (2005b). *Service-Oriented Architecture Concepts, Technology, and Design.* Prentice Hall.

Filter UK. (2006). *Bluecasting from Filter*. Retrieved June 25, 2008, from http://www.bluecasting.com/home.html

Hansen, D. M. (2007). *SOA, Using Java Web services.* Indiana: Prentice Hall.

Hopkins, G. W. (2000, May 4). *sportsci.org.* Retrieved July 15, 2008, from www.sportsci.org: http://www.sportsci.org/jour/0001/wghdesign.html

Horstmanshof, L. (2004). Using SMS as a way of providing connection and community for first year students. In C. M.-D. R. Atkinson (Ed.), *21st ASCILITE Conference*, (pp. 423-427). Perth.

Howell, D. C. (1989). *Fundamental Statistics for the Behavioral Sciences* (2nd Edition ed.). PWS-KENT Publishing Company.

Hull, R., & Su, J. (2005). Tools for Composite Web services. *SIGMOND. 34.* ACM.

International Telecommunication Union. (2008, July 15). *Global ICT developments*. Retrieved August 25, 2008, from http://www.itu.int/ITU-D/ict/statistics/ict/index.html

Jones, M., & Marsden, G. (2006). *Mobile Interaction Design.* John Wiley & Sons, Ltd.

Lam, W., & Shankararaman, V. (2007). *Enterprise Architecture and Integration: Methods, implementation. and Technologies.* New York: Information Science Reference.

Londondev Business Solution. (2006). *BroadTooth Homepage - the Broadtooth broadcasting system*. Retrieved July 17, 2008, from http://www.broadtooth.com

May, P. (2001). *Mobile commerce: opportunities, applications and technologies of wireless business.* Cambridge: Cambridge University Press.

Midletsoft. (2006). *Jellingspot.com-Aren't you Jellingspot enabled?* Retrieved July 15, 2008, from http://www.jellingspot.com

Musisi, N. B., & Muwanga, N. (2003). *Makerere University In Transition 1993-2000.* Kampala: Fountain Publishers.

OSGiAlliance. (2005, November). *About the OSGi service platform: Techinical whitepaper.* Retrieved from http://www.osgi.org/documents/collateral/TechnicalWhitePaper2005osgi-sp-overview.pdf

Paek, T., Agrawala, M., Basu, S., Tayoma, K., Ducker, S., Kristjansson, T., et al. (2004). Toward universal mobile interaction for shared displays. *Computer supported collaborative work.* Chicago, Illinois: ACM.

Papazoglou, M. P., & Georgakopoulos, D. (2003). Service-oriented computing. *ACM, 46* (10), 24-28.

Preece, I., Rogers, Y., & Sharp, H. (2002). *Interaction Design- Beyond Human-Computer Interaction.* John Wiley & Sons.

Russell, D. M., Drews, C., & Sue, A. (2002). Social Aspects of using large public interactive displays for collaboration. *UbiComp*, (pp. 229-236).

Sawyerr, A. (2004). *Challenges facing African Universities: Selected issue.* Accra, Ghana: Association of African Universities.

Schmelzer, R. (2002, October 24). Retrieved October 20, 2008, from "Breaking XML to Optimize Performance."ZapThink: http://searchwebservices.techtarget.com/originalContent/0,289142, sid26_gci858888,00.html

Singh, P. M., & Huhns, N. M. (2005). *Service-Oriented Computing Semantics, Processes, Agents.* John Wiley & Sons, Ltd.

Srinivasan, L., & Treadwell, J. (2005). *An Overview of Service-oriented Architecture.* HP Software Global Business Unit.

Sun Microsystems. (2005, June 14). *The java Web services Developer Pack 1.6.* Retrieved October 20, 2008, from http://www.j2ee.me/webservices: http://www.j2ee.me/webservices/doc/1.6/tutorial/doc/JavaWSTutorial.pdf

The Bluetooth Special Interest Group. (1999, December 1). *"Specification of the Bluetooth System: Part B Baseband Specification".*

Theotokis, S. A., & Diomidis, S. (2004, December). A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys , 36* (4), pp. 335-371.

Treadwell, J. (2005, January 25). Open Grid Services Architecture Glossary Of Terms. *Open Grid Services Architecture WG, Global Grid Forum* .

Tutsch, D. (2006). *Performance Analysis of Network Architectures.* Berlin: Springer.

W3C. (2004, Febraury 11). Retrieved 11 21, 2008, from http://www.w3.org/TR/ws-arch/

Whitman, E. M., & Woszczynski, B. A. (2004). *The Handbook of Information Systems Research.* Hershey: IDEA GROUP PUBLISHING.

Woodings, R., Joos, D., Clifton, T., & Knutson, D. C. (2002). Rapid Heterogeneous Connection Establishment: Accelerating Bluetooth Inquiry Using IrDA. *In Proceedings of the Wireless Communications and Networking Conference(WCNC2002)* , 342–349.

APPENDIX A

A sample structured text document.

E-du Box: Educational Multimedia with Tangible-Enhanced Interaction

André Wilson Brotto Furtado
Cento de Informática - UFPE
Rua Prof. Luis Freire, s/n
B.P. 7851 - 50732-970
Recife PE Brasil
+55.81.91827818
awbf@cin.ufpe.br
Taciana Pontual Falcão
Cento de Informática - UFPE
Rua Prof. Luis Freire, s/n
B.P. 7851 - 50732
Recife PE Brasil
+55.81.86025607
tacianapontual@gmail.com
Alex Sandro Gomes
Cento de Informática - UFPE
Rua Prof. Luis Freire, s/n
B.P. 7851 - 50732-970
Recife PE Brasil
+55.81.21268430
asg@cin.ufpe.br
Carlos Eduardo Monteiro Rodrigues
Cento de Informática - UFPE
Rua Prof. Luis Freire, s/n
B.P. 7851 - 50732-970
Recife PE Brasil
+55.81.88245862
cemr@cin.ufpe.br
Roberto Sonnino
Escola Politécnica
Universidade de São Paulo
Av. Prof. Luciano Gualberto, travessa 3 n° 380 - CEP: 05508-900
São Paulo - SP - Brasil
+55.11.47276746
robertos@gmail.com

ABSTRACT
Media resources usage has significant impact on children literacy in the first school years in Brazil [5]. Computer software and tangible interfaces can help engage pupils in effective learning activities. Tangible interfaces built with familiar objects of our everyday lives such as wood and tissues are well accepted by pupils. In this work, we detail our design and evaluation of e-du box - an educational, authoring and sharing multimedia platform including a tangible companion that provides feedback for users. We employed a participatory design process based on providing supports intended to help

children engage in different tasks. We could elicit a list of design guidelines for this specific application. We discuss our experience with this design approach and explore its implications.

Categories and Subject Descriptors
H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces. H. 5. 1 [Multimedia Information Systems]
General Terms
Design, Human Factors.
Keywords literacy, tangible interaction, educational software, participatory design, social technology

1. INTRODUCTION

Literacy is a process that promotes socialization, since it establishes new types of symbolical exchanges among individuals, access to cultural assets and facilities offered by social institutions. It is also a key to the concept of citizenship and to society development in general, as well as is strongly related to health, better income and, therefore, life quality. Recent studies from UNESCO, however, present some alarming facts related to literacy: more than one fifth of the worldwide population is illiterate. Over 771 million people (15 years old or more) are unprovided of basic reading, writing and calculation skills [4]. In Central Africa, for example, there is an impressive illiteracy rate of 80%. Brazil, the country where the first prototypes of the product presented in this work have been developed and tested, is not an exception. It is one of the 12 countries where three quarters of the illiterate adults live. More than 15 million Brazilians are not able to read what is written in their own national flag. If definitions such as functional, digital and social illiteracy are taken into account, the results are even worse [19].

In order to use the potential of technology to revert such statistics, some companies, universities and non-profit organizations are developing projects to create low-cost computers for educational purposes [12]. One characteristic shared among those projects is the fact that they are just simplified versions of regular personal computers, still keeping their appearance and user interface, while carrying with them some of the costs of full-sized computer's solutions. Furthermore, research with educators, pedagogues, students and other stakeholders revealed additional limitations and restrictions of current literacy approaches, including: the lack of possibilities for educators to customize learning content according to each student context, preferences and difficulties; a high dependency level of the learning process on classrooms and the presence of educators; the absence of a deep parental

DIS 2008, February 25-27, 2008, Cape Town, South Africa.

involvement in the learning process; the bureaucratic and nonautomated way in which students' progress is assessed; the lack

of incentives to make content created by different educators around the world to be easily published and shared and, finally, the sub-exploration of technology to provide a real intuitive and usable interface, which makes digital approaches to be seen as complex and hard to use.

Taking such issues into account, we present the e-du box, a low-cost and flexible hardware/software edutainment platform, conceived with focus on literacy. It empowers educators with the necessary elements to provide rich (yet localized) multimedia content to students. The solution allows learners to experience content created by their own educator in an intuitive and stimulating way. E-du box provides a rich multimedia experience, enhanced by tangible interaction. The main input device is a special pen-shaped mouse that vibrates according to some situations defined by the educator. Feedback is also provided by a tangible, interactive and animated e-du agent, who is able to move and speak to students. The device can be connected to ordinary TV sets, already existent in the great majorities of homes, therefore decreasing the overall cost of implementing the solution.

The design process of e-du box is distinct as we explicitly modeled the interface design after analyzing usage in human experience dealing with a low fidelity prototype application. We conducted qualitative field studies to identify teachers and students necessities on constructing and interacting with our prototype. We continually improve the interactions provided to users as we could identify lacking and exceeded features.

In the next section we discuss research and tangible interfaces usage in education. We then present the design of e-du box and the findings of our evaluation of the tools in use. We conclude with discussion of future evolutions of this solution.

2. TANGIBLE INTERFACES IN EDUCATION

Tangible user interfaces (TUIs) seek to change the traditional I/O paradigm of computer systems, creating possibilities of interaction that bring together the digital and physical worlds [18] apud [7] and try to be more natural to human beings. For that purpose, innovative I/O devices are being proposed, in many cases making use of concrete objects of our everyday lives [14].

Zuckerman et al. [20] propose the following classification for TUIs:

1. TUI as input, GUI (graphical user interface) as output: a tangible interface is used as the input device of a computer system, and the output is shown on a screen.
2. Output projected on a TUI: the output of a computer system is not projected on a GUI, but on a tangible interface.
3. Immersive environments: TUIs are interfaces through which users interact with a pervasive computing system.
4. Computing power embedded in physical objects: the TUI works as the input and output devices, with no GUIs involved.

The innovative forms of interaction provided by tangible interfaces make them popular in the Education field. TUIs open new possibilities in didactic practices, extending students' learning experience [11] by putting together advantages of digital data (like updating capabilities) and physical aspects of tangible resources [13].

Zuckerman et al. [20] also cite some advantages brought by tangible interfaces in Education:

- Sensorial engagement: children learn in natural ways, using different senses (touch, vision, and hearing) in a constructive process which increases retention of learning content.
- Accessibility: tangible interfaces provide more options of interaction to include children with special needs.
- Group learning: with tangible interfaces, discussions and collaborative work are usually encouraged.

Hoyles & Noss [2] apud [10] add that TUIs bring fun, exploration, reflection, imagination, creativity and collaboration to learning.

In educative systems, the focus is not technology, but the interaction and its effects [10]. Interfaces should direct user attention to the object of the learning activity [7]. If students need to concentrate on how to manipulate the interface instead of thinking of the concept to be studied, learning will probably be less effective [9].

3. DESIGN PROCESS

The development of e-du box was based on the results of a qualitative field research made in Recife, Brazil. The first step was to understand the context of the educational field, identify teachers' and students' needs related to literacy and detect opportunities for technology to be used for their benefit. The output from this initial research was key to the ongoing project development efforts. The collected feedback was properly compiled into the guidelines presented in Section 3.2.

We were oriented by a social constructivist framework in our design decisions [1] perceiving reflexive acting and social interaction as part of the origins of human development.

3.1 Participants

We visited schools and interviewed their teachers, coordinators and directors about their teaching practices. We also had informal conversations with specialists from universities, such as pedagogues and psychologists. A positive feedback was given as for the use of technology: both students and teachers became enthusiastic about it. Furthermore, the team received valuable feedback regarding assessments content. Our approach was changed from the learning of letters to exercises which explore the meaning of words in a social context.

Those interactions allowed us to exchange ideas on the conception of e-du box and led us to state some guidelines for the development of the product. We describe those guidelines in the following section.

3.2 Guidelines

1. Contextual on-demand educative content: today's approaches have considerable limitations for educators to add and change educative content related to specific difficulties, preferences and day by day activities of each student. For example, a teacher reported that she attempted to use, in the northeast region of Brazil, educative content (books, exercises, among others) created in the southeast region of the country. However, she was not successful, since such content was based on cultural issues of the southeast region which were not so meaningful to northeast students. Another issue which could be addressed here is the difference among students' level and abilities in a same class. In Brazil, it is quite common to have students who

can read together with illiterate ones. It would be, therefore, very convenient for the teacher to have the possibility of
141
adapting activities for different levels according to the students necessities.
2. Technology as a mean, not a goal: as mentioned before, technology is not the focus in educational artifacts, but a mean to improve the learning process. Another point is that teachers must be able to deal with technology and adapt their methods to integrate the new artifacts in class. In Brazil, many teachers are still resistant to use technology, mainly for fear of being unable to manage it in class. In this context, a major requirement is that technology should be easy and simple to set up, use and customize.
3. Focus on student motivation: the majority of students' assessments use pencil and paper as the underlying media. Such fact, combined with the repetitive nature of some assignments, contribute to the loose of motivation. On the other hand, activities that are related to richer media and games (not necessarily digital games) are the ones in which students get more deeply engaged. Hence, the success of future educative approaches requires that they move the student role from static to interactive.
4. Deeper parental involvement: schools reported that today's educational system is still highly dependent on classrooms and the presence of educators. Some pedagogues recognized that many of the exercises assigned to students do not provide the opportunity for a deeper interaction with parents at home or friends outside school. Therefore, learning is not stimulated beyond classrooms. Internet at home still does not solve the problem, since it is not targeted (yet) to the illiterate.
5. Aided continuous and reflexive assessment process: the way students' progress is assessed today is manual and done case by case. This can make the assessment process bureaucratic, repetitive and error-prone. However, we were able to identify that students' assessment, even in the literacy domain, can be richer and more automated. For example, the traditional student portfolio, which is kept by the teacher to gather all concluded exercises along the year, could be automatically and gradually built once based in digital assessments. Furthermore, digital technologies can make it possible not only to view the result of an assessment solving but also to check each step carried out by the student.
6. Community reuse: there is a lack of possibilities to make content creation and sharing by different educators around the world an easy task. In other words, educators have few means to exchange educative assets aided by tool support, efficient searching and automation. One of the reasons is that such assets are not digital, and even if they were, there is no easy way to create and maintain them. However, interviews revealed that teachers would be interested in taking part in online assessment exchange communities to share knowledge and resources.
7. Articulation with the private initiative: many successful social projects are carried out and concluded with the collaboration of the private initiative. Some of the interviewed stakeholders believe that literacy is a field where the articulation with the private initiative can arise indeed,

especially due to the urgent investments needed to revert current illiteracy statistics.

3.3 E-du Box

With the aforementioned principles in mind, we have conceived the e-du box, a low-cost and flexible hardware/software educational platform. E-du features go beyond literacy purposes, addressing both learners' personal skills and their understanding about specific concepts. At the same time, it also empowers educators with an interesting tool to build rich (yet localized) assessments situations based on multimedia content, as well as to assess students' progress in a more automated way. The high-level platform components are presented in Figure 1.

In the client side, the e-du box is a computing device that can be connected to ordinary TV sets, already existent in the great majorities of homes, which would work as the output display. The graphical interface shown on the TV screen resembles a notebook (a traditional one, not the laptop computer) (see Figure 2). The digital notebook layout, such as its cover, can be different for each user. Actually, the whole user interface can be customized.

142

Figure 6. The playful interface of Edupedia

The activities available for students can be pre-built in the application or deployed on-demand by the educator, through a personal computer (represented by the educator's computer in Figure 1). Using a Bluetooth connection, educators can synchronize with students' e-du boxes, collecting information from them, and be aided by a tool called E-ducator to have on-demand graphics and reports to assess students' progress automatically. The way a student solved a specific assessment can be reproduced as well. In this way, the educator can monitor the progress of a whole group of students in a centralized and automated fashion.

Once with enough information to define subsequent or corrective learning strategies, educators can develop and deploy to students' e-du boxes new multimedia interactive exercises. To accomplish that, they can launch a simple yet powerful WYSIWYG authoring tool, called E-ducreator that runs in the educator's computer, which is used to create lessons. This tool makes it possible to program the input and output capabilities of the e-du box through an intuitive interface. An example is shown in Figure 4. This allows dictations, reading, drawing and many other more advanced school activities to be presented a much more appealing way.

The possibility of updating the students' application addresses an important requirement identified by educators: the student learning process should not be dissociated from reality and current facts. For example, students should not only be able to read and write the word "war", but also to understand where and why wars are happening in the world and what is the consequence of such fact to their lives.

Besides deploying the assessment to an e-du box, educators can choose to publish and share their lessons (knowledge and resources) through Shar-e-du, an online educative assessments repository, shown in Figure 5. Items in the repository can be searched by tags in two different ways: from within the authoring tool or through a web portal. Such a repository

provides new Web 2.0 capabilities [8] to the edutainment domain, empowering educators with new collaboration and reuse opportunities.

Students are also able to launch the Edupedia (Figure 6), a "learning encyclopedia", which presents concepts together with related videos, pictures and text, whose complexity level is based on the acquired skills of the student. Concepts related to the current concept are presented through a graph in the left page, inviting the student to explore new content and, consequently, to learn more.

Users interact with the interface by using a pen-shaped mouse [3]. Such experience is intended to be as intuitive and straightforward as the use of a pencil and a sheet of paper, not requiring any previous computer experience or training. However, due to the digital nature of e-du, the interaction level may be much richer than the pencil/paper experience. Moreover, the pen would be equipped with vibrating mechanisms, similar to a force-feedback joystick, in order to better enrich the student interaction.

Figure 3. The Draw Together activity view

Figure 4. E-du creator authoring interface for building an activity

Figure 5. The sharing interface of Shar-e-du

143

Figure 8. Experiments scenario

The client side is also composed by a real, interactive tangible agent, which can be customized. The agent is able to be in motion and speak to students, in order to provide real time feedback and motivate the self-learning process. The communication between the agent and the e-du box is wireless, therefore it can also be moved to any nearby place. A real e-du box usage scenario, with the TV, the e-du box, the pen and the agent, is shown in Figure 7.

4. EVALUATION

A first prototype was developed and brought to schools to be informally evaluated by teachers and students. The focus in this first phase was on the content to be provided to students (e.g., assessments, videos, games), as well as the provided graphical interface. How well students and teachers would accept the technology and deal with it was further investigated. Finally, a "manually operated" version of the interactive agent was presented to teachers, in order to verify the product usability. A second, higher-fidelity prototype was then developed and more formal tests were made with students. Such an evolved prototype matches the features set described in Section 3.3. The focus was on validating the tangible interface and the platform as a whole. In Section 4.2 we present our analysis of the usability level regarding task effectiveness and learnability.

4.1 Participants and context

Tests were performed in a private school in Recife, Brazil. Students belonged to middle class and were familiar with computers. They were aged 5-6 years and were starting to get to know the alphabet and perform literacy activities. The prototype was tested using two monitors side by side. Six students took part in the experiment, working in pairs. The experiment scenario is shown in Figure 8.

4.2 Activities

In this section, we describe two specific tasks evaluated with users: "memory game" and a "circle the picture" activity.

Memory Test Game: in such a game, two students, using the same computer (eBox) and output display device (a monitor or television), are presented a board with six cards. All cards are turned backwards. The game is played in turns. The first student chooses a card, by clicking on it. The card is flipped and then displays either a word or a drawing. The student's goal is to find the matching card: if the original card contains a drawing, the matching card contains the word describing that drawing (such as "dog" or "duck", for example). A point is awarded for each successful matching. In the case of an unsuccessful matching, the selected cards are flipped backwards again. Both student scores are displayed in the left page of the screen. After a student selects his second card in a turn, the other student owes the next turn, no matter if the previous student was successful or not. The students share a same pen-shaped mouse, in turns, as input device. The external agent congratulates the students when a successful match is discovered and says "Not that, try again" when the card pair matching is unsuccessful. Finally, when all cards are flipped, the game ends and the agent congratulates the winner.

Circle picture: in this activity, two students, again using the same computer (eBox) and output display device, are shown some pictures on the screen and are asked to circle the picture whose name starts with a particular letter, also shown on the screen. The students also share a same pen-shaped mouse, in turns, as input device, and have feedback from the tangible agent.

4.3 Data collection and analysis

During evaluation sections, interactions were registered on video for further transcription and qualitative data analysis. A camera was placed behind students capturing their conversation, movements, negotiation protocols and the computer screen display. In this condition, we were able to track the mediation aspects of the interface in the collaborative problem solving situations.

We used collaborative data collection and analysis [6] to obtain accurate results from data faster. In so doing, more interaction design cycles could exist and in consequence more evaluation moments.

5. RESULTS

We present the results of our evaluation through categories derived from the data analysis.

5.1 E-pen is not really a pen

Different from our previous point of view, the pen-shaped mouse is not directly associated to a real pen by students. Students had to be taught how to use the pen as input device, and needed some help to make it work properly. The main problems children faced while interacting with the pen were: holding the pen in such a way that it would touch the table surface and work properly as a mouse; move the pen slowly enough to produce a specific drawing on the screen; synchronize the movement of the pen with the pressing of the buttons to select icons or produce a drawing. However, despite

Figure 7. A real e-du box usage scenario

144

those problems, children were eager to learn how to use the pen and, with that, perform the tasks. During the experiment, some students declared it was quite difficult to circle a picture or make a drawing using the pen, but after some training children said: "I know how to do, it's easy, it's easy" and started teaching their mates.

5.2 High tolerance threshold

Despite some technical problems and functional limitations of our prototype, students maintained their engagement during the whole experiment. They were patient and good-humored, showing no negative reactions while waiting to interact with our system.

5.3 Affective aspects in agent-user interaction

Children were involved with the tangible agent, chatting with it ("hello, what's your name?"), holding its hand, asking permission to touch it and talking about it ("oh! It´s dancing!"). Each time the agent gave some feedback, students looked at it laughing with surprise and joy.

5.4 Captivating interface and activities

Children were motivated and engaged with the interface and activities proposed. They showed interest, pleasure and care while performing the tasks.

6. DESIGN IMPLICATIONS

Reflecting our previous results, we observed some implications in our design process. Two of them are related to the product (6.1 and 6.2) and two others (6.3 and 6.4) with the evaluation methodology.

6.1 Feedback edited by teacher

As situations can be created by teachers, another customization level is related to how the tangible agent can provide movement and audio feedback related to the kind of problem and the level the students are. We could reflect in terms of scaffolding strategies implemented using this part of the interactive system to be closer related to student's comprehension and involvement in the practice.

6.2 Input tool suitable for young children

When developing a product for young children, we must be very careful when considering the specific input tool. Youngsters may not have developed an accurate motor coordination yet and need a suitable tool for proper and easy interaction. The pen-shaped mouse proved to be a good idea, yet it needs to be improved in order to better react and capture user's movements.

6.3 Considering aesthetics and affective aspects

We noted obvious aesthetics and affective evidences on the usage of our system by young children. We must include the quality and type of the material used to construct the tangible agent as a variable to understand the relation with user's reactions and acceptance.

6.4 Understanding interaction as an overall communication process

During the evaluation phase, it is important to understand the overall interaction involving different users in the situation and the teacher previous and asynchronous interaction through the activity planning and distribution and further results collection

and evaluation. This complete series of interactions must be conceived to be as continuous as possible in terms of communications over time and inter situations. These considerations have impacted on the way we'll plan the future observations and data analysis in the evaluation phases.

7. DISCUSSION

Several tangible interfaces have already been proposed for Education. One of them is the Tangible Interface for Collaborative Learning Environments (TICLE) [16], a platform which uses computer vision techniques to track concrete objects and map their movements to personal computers. Children interact with concrete mathematical games (such as Hanoi Tower and tangram [17]) while the system maps their actions to the computer screen, showing their evolution and offering help to guide them towards the solution.

Scarlatos & Scarlatos [17] also developed mathematical mats: SmartStep and FloorMath. The mats have sensors to detect children's movements on top of them, and are connected to a personal computer which shows a virtual representation of the mat and the activities that should be done by the student. The mats use physical activity to practice math concepts like counting and basic operations.

In Brazil, the educational tables of the company Positivo (www.positivoinformatica.com.br) use concrete didactic materials for children to interact with the personal computer. The activities associated with the tables relate to several knowledge areas and stimulate coordination, visual perception and logical reasoning. Activities are structured according to students' age and level.

Another example is the I/O Brush [15], a drawing tool in the shape of a common paintbrush, but with an embedded camera and touch sensors. Such devices allow users to capture colors and textures of surfaces and reproduce them on the drawing canvas (consisting of a large touchscreen and a back projection screen).

As for external interactive agents (companions), a lack of solutions targeted at the educative domain was observed. Some currently existent agents actually execute simple actions, such as dancing or blinking when an instant messenger contact becomes online. Other agents are able to pronounce some words and small phrases, but none of them are connected to software systems as part of a broader educative platform.

The products cited above can be classified in the category of "TUI as input, GUI as output" (see Section 2), i.e., input is done through a tangible interface (concrete math games, mats, didactic concrete materials, paintbrush) and output is shown on a separate screen.

As for e-du box, input is done through a special pen and output occurs through three different forms. One of them is the vibration of the pen, which represents a tactile feedback - the pen can therefore be considered an input and output device. Another answer of the system is given through the external agent (a doll) that interacts with the users as a companion to guide them through the activities. Besides being tangible, the companion has a playful aspect, making learning more fun for children and letting them more involved. Finally, the system shows the educational activities on a separate graphical

interface (television screen). Therefore, e-du box goes beyond the approach of "TUI as input, GUI as output", broadening the range of interactions between the child and the system and providing a richer and more dynamic learning environment.
145
According to the guidelines presented in Section 3.2, we defined a group of parameters (ranging from 1 to 5) to analyze e-du box and the related products presented in this section. These parameters are: interactivity (INT), connectivity among students (CAS), parental involvement (PIN), audience reach versus cost rate (ACR), personalization (PER), updating capabilities (UCA), openness to external private initiatives (OPI) and community collaboration (CCO). Table 1 presents the results of this analysis.
Table 1. Comparative analysis: E-du Box and related products
TICLE Mats Tables I/OBrush E-du
INT 5 5 5 5 5
CAS 3 1 5 2 3
PIN 3 3 1 3 4
ACR 2 2 3 3 3
PER 1 1 4 1 5
UCA 1 1 1 1 5
OPI 4 4 1 4 4
CCO 3 1 1 3 4
Total 22 18 21 22 33
We see that, although E-du Box does not reach the maximum grade (5) in all parameters, it is the product - of the group analyzed - which better addresses the needs we consider important to be satisfied to promote collaborative learning in situ.
Besides those features, educational effect of e-du box involves impacts on a series of individual and group interaction styles, and can also be analyzed according the aspects presented in Section 3.2.
8. FUTURE WORK
The real impact of e-du box in the learning process can be assessed in a long-term basis, by putting the system in use in some classrooms and comparing students' performance in school with and without it. However, the enthusiasm and interest of students and teachers during the experiments show that there is a good scenario for e-du box to be successful in classrooms.
We believe some features could be added to enhance users' interaction with e-du box applications: speech and handwriting recognition could make the interaction more natural. Speech recognition could allow children to actually "talk" to the external tangible agent, which could increase their engagement in the activities and make the interaction with the agent richer than pure feedback. Handwriting recognition would allow children to use the pen-shaped mouse in a way closer to the real use of a pen.
Our work could also be easily expanded to cover other content topics than literacy. Mathematics or Science activities, for example, would broaden the range of possible situations of use of e-du box by educators.
We believe e-du box could also be adopted for distance

education. Since we already provide interfaces for the educators to create activities, feed students' equipments with them and collecting results afterwards, changes in the way of communication among users' equipments and, possibly, adaptations to enhance awareness could make e-du suitable for distance education.

Regarding the cost effectiveness of e-du in developing countries, practical and economic aspects of schools in those countries, such as in Brazil, can be taken into account in order to improve the solution adoption. For example, it is possible to make use of already existent infra-structure scenarios, such as low-cost laptops (as those from the One Laptop Per Child initiative or the Intel Classmate PC) to deploy e-du. Other possibility is to deploy e-du to computers belonging to the donation market scenario, i.e., old computers donated to schools by companies and non-profit organizations. Since the processing power of such machines matches the current platform capabilities (eBox), this would not be a constraint. Other platforms such as mobile phones, digital TV, and UltraMobile PC could also be explored. However, we believe a careful and deeper analysis of users interaction in each case must be made before actual development of adaptations is performed.

9. CONCLUSION

In this paper, we have presented the design of e-du box - an educational, authoring and sharing multimedia platform including a tangible companion that provides feedback for users. We have also presented a field study which contributed to our understanding of how pupils engage in learning tasks and cooperate in synchronous situation through the interface, besides generating a set of guidelines used in the conception of our product. We could progress in understanding the interactions realized through this interface. We could observe the main interaction possibilities and the main limitations of our concepts and design process.

10. ACKNOWLEDGMENTS

We would like to thank the Center of Informatics of the Federal University of Pernambuco (CIn - UFPE), the researchers we interviewed and the students, teachers and all school staff that collaborated in some way with our research.

11. REFERENCES

[1] Bruner, J. (1990). Acts of Meaning. Cambridge, MA: Harvard University Press.

[2] Hoyles, C. & Noss, R. Playing with (and without) words. In Proceeedings of the Seventh European Logo Conference (Eurologo'99), pages 18-29, Bulgaria, 1999.

[3] i-Pen: Presentation Digital Pen / Optical Pen Mouse. Available at: http://www.zyonshop.com/product/ipen.htm. Accessed on: Aug 2007.

[4] MEB - Basis Education Movement, "News about literacy - data from Unesco". Available at: http://www.meb.org.br/noticias/unescoalfabetizacao. Accessed on: Jun 2007.

[5] MEC, Brazil Education Ministry, Relatório SAEB 2001. Available at: http://www.inep.gov.br/basica/saeb/. Accessed on: Jan. 2007.

[6] Millen, D. R. Rapid ethnography: Time deepening

strategies for HCI field research. In Symposium on Designing Interactive Systems - DIS'00, New York. ACM, 2000.
[7] O'Malley, C., Fraser, D. S.: Literature Review in Learning with Tangible Technologies. NESTA Futurelab (12), 2004. 146
[8] O'Reilly, T. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. 2005. Available at:http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html. Accessed on: Jul. 2007.
[9] Oppenheimer, T. The computer delusion. The Atlantic Monthly, 280(1):45-62. 1997.
[10] Price, S., Rogers, Y., & Scaife, M. Using tangibles to promote novel forms of playful learning. Interacting with Computers, 15(2):169-185. Special Issue on Interaction Design and Children. 2003.
[11] Proctor, R. M. J., Baturo, A. R., & Cooper, T. J. Integrating concrete and virtual materials in an elementary mathematics classroom: a case study of success with fractions. In Proceedings of Seventh World Conference on Computers in Education, volume 8, pages 87-92. 2001.
[12] Quick guide to low-cost computing devices and initiatives for the developing world. Available at: http://www.infodev.org/en/Publication.107.html. Accessed on: Aug. 2007.
[13] Raffle, H., Parkes, A., & Ishii, H. Topobo: A constructive assembly system with kinetic memory. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'04), Austria. 2004.
[14] Resnick, M., Maryin, F., Berg, R., Boovoy, R., Colella, V., Kramer, K., et al. Digital manipulatives: new toys to think with. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'98), pages 281-287, USA. 1998.
[15] Ryokai, K., Marti, S., & Ishii, H. I/O Brush: drawing with everyday objects as ink. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'04), Austria. ACM Press. 2004.
[16] Scarlatos, L. L., Dushkina, Y. & Landy, S. TICLE: A tangible interface for collaborative learning environments. In Extended Abstracts of the SIGCHI Conference on Human Factors in Computing Systems (CHI'99), pages 260-261, USA. 1999.
[17] Scarlatos, T. & Scarlatos, L. Tangible Math Applications. Available at: http://www.cs.sunysb.edu/~tony/research/Math.pdf. Accessed on: Aug. 2006.
[18] Ullmer, B. & Ishii, H. Emerging frameworks for tangible user interfaces. IBM Systems Journal, 39(3/4):915-931. 2000.
[19] UNESCO. Available at: http://unesco.org. Accessed on: Aug 2007.
[20] Zuckerman, O., Saeed, A., &Resnick, M. Extending tangible interfaces for Education: digital Montessoriinspired manipulatives. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'05), pages 859-868, USA. ACM Press. 2005.

APPENDIX B

Abstract extract.

CSC500W
E-du Box: Educational Multimedia
Media resources usage has significant impact on children literacy in the first school years in Brazil [5]. Computer software and tangible interfaces can help engage pupils in effective learning activities. Tangible interfaces built with
familiar objects of our everyday lives such as wood and tissues
are well accepted by pupils. In this work, we detail our design
and evaluation of e-du box ? an educational, authoring and sharing multimedia platform including a tangible companion that provides feedback for users. We employed a participatory
design process based on providing supports intended to help children engage in different tasks. We could elicit a list of
design guidelines for this specific application. We discuss our

APPENDIX C

A course site device XML registration file

```
<?xml version="1.0" encoding="UTF-8" ?>
<macUserLists>
 <course>
 <name>CSC500W</name>
     <userMacAddress>0016B8E502DB</userMacAddress>
     <userMacAddress>0800289472F6</userMacAddress>
     <userMacAddress>0064B914AABC</userMacAddress>
     <userMacAddress>080028952F16</userMacAddress>
     <userMacAddress>00166F07BCF9</userMacAddress>
     <userMacAddress>080028923F3E</userMacAddress>
     <userMacAddress>08002893A52F</userMacAddress>
     <userMacAddress>080028924F9E</userMacAddress>
     <userMacAddress>080028947CEA</userMacAddress>
     <userMacAddress>0800289544F6</userMacAddress>
     <userMacAddress>080028947CEA</userMacAddress>
     <userMacAddress>08002895D7A5</userMacAddress>
     <userMacAddress>08002893BB22</userMacAddress>
 </course>
 </macUserLists>
```

APPENDIX D

Raw data for Off-peak usage experiment.

Device Name	Tb1	Ta1	Tb1-Ta1	Tb2	Ta2	Tb2-Ta2
Mercury	1.18692E+12	1.18692E+12	0.39661667	1.19E+12	1.19E+12	0.419267
Mars	1.18692E+12	1.18692E+12	0.9375	1.19E+12	1.19E+12	0.7737
Saturn	1.18692E+12	1.18692E+12	2.4263	1.19E+12	1.19E+12	1.564317
Uranus	1.18692E+12	1.18692E+12	2.15208333	1.19E+12	1.19E+12	1.291933
Jupiter	1.18692E+12	1.18692E+12	1.13828333	1.19E+12	1.19E+12	2.62865
Pluto	1.18692E+12	1.18692E+12	4.41536667	1.19E+12	1.19E+12	1.956517
Moon	1.18692E+12	1.18692E+12	4.1203	1.19E+12	1.19E+12	1.797133
Sun	1.18692E+12	1.18692E+12	3.05025	1.19E+12	1.19E+12	5.100783
Asterix	1.18692E+12	1.18692E+12	2.7125	1.19E+12	1.19E+12	4.421883
		Average Response Time(Minutes)	2.37213333			2.217131

Tb3	Ta3	Tb3-Ta3	Tb4	Ta4	Tb4-Ta4	Tb5	Ta5	Ta5-Ta5
1.19E+12	1.19E+12	0.433333	1.19E+12	1.19E+12	0.264583	1.19E+12	1.19E+12	0.370567
1.19E+12	1.19E+12	0.722917	1.19E+12	1.19E+12	1.074483	1.19E+12	1.19E+12	1.14635
1.19E+12	1.19E+12	1.489317	1.19E+12	1.19E+12	1.43255	1.19E+12	1.19E+12	5.039583
1.19E+12	1.19E+12	1.205717	1.19E+12	1.19E+12	4.5237	1.19E+12	1.19E+12	1.01615
1.19E+12	1.19E+12	2.743483	1.19E+12	1.19E+12	5.688017	1.19E+12	1.19E+12	1.36615
1.19E+12	1.19E+12	2.046867	1.19E+12	1.19E+12	0.759633	1.19E+12	1.19E+12	1.790367
1.19E+12	1.19E+12	1.355983	1.19E+12	1.19E+12	1.146083	1.19E+12	1.19E+12	3.903133
1.19E+1	1.19E+1	4.91041	1.19E+1	1.19E+1	3.37603	1.19E+1	1.19E+1	4.5112

2	2	7	2	2	3	2	2	
1.19E+12	1.19E+12	4.590617	1.19E+12	1.19E+12	1.093483	1.19E+12	1.19E+12	1.1573
		2.166517			2.150952			2.255644

Tb6	Ta6	Tb6-Ta6	Tb7	Ta7	Tb7-Ta7	Tb8	Ta8	Tb8-Ta8
1.19E+12	1.19E+12	0.29115	1.19E+12	1.19E+12	0.270833	1.19E+12	1.19E+12	0.291917
1.19E+12	1.19E+12	1.125	1.19E+12	1.19E+12	1.077617	1.19E+12	1.19E+12	1.083333
1.19E+12	1.19E+12	1.40025	1.19E+12	1.19E+12	1.430483	1.19E+12	1.19E+12	1.445833
1.19E+12	1.19E+12	2.637233	1.19E+12	1.19E+12	6.215883	1.19E+12	1.19E+12	6.05495
1.19E+12	1.19E+12	4.711717	1.19E+12	1.19E+12	2.2414	1.19E+12	1.19E+12	3.885417
1.19E+12	1.19E+12	0.739317	1.19E+12	1.19E+12	0.838283	1.19E+12	1.19E+12	0.817983
1.19E+12	1.19E+12	5.758067	1.19E+12	1.19E+12	3.073433	1.19E+12	1.19E+12	3.7224
1.19E+12	1.19E+12	2.390633	1.19E+12	1.19E+12	3.422133	1.19E+12	1.19E+12	0.70105
1.19E+12	1.19E+12	1.016933	1.19E+12	1.19E+12	1.137233	1.19E+12	1.19E+12	1.045317
		2.230033			2.1897			2.116467

Tb9	Ta9	Tb9-Ta9	Tb10	Ta10	Tb10-Ta10
1.19E+12	1.19E+12	0.2651	1.19E+12	1.19E+12	0.245833
1.19E+12	1.19E+12	1.09635	1.19E+12	1.19E+12	1.005733
1.19E+12	1.19E+12	1.360933	1.19E+12	1.19E+12	1.3599
1.19E+12	1.19E+12	1.6789	1.19E+12	1.19E+12	4.745317

1.19E+12	1.19E+12	3.8198	1.19E+12	1.19E+12	1.2513
1.19E+12	1.19E+12	4.960417	1.19E+12	1.19E+12	1.985417
1.19E+12	1.19E+12	1.00105	1.19E+12	1.19E+12	4.905467
1.19E+12	1.19E+12	1.406783	1.19E+12	1.19E+12	3.57395
1.19E+12	1.19E+12	2.661983	1.19E+12	1.19E+12	1.3599
		2.027924			2.270313

APPENDIX E

Raw data for peak usage experiment.

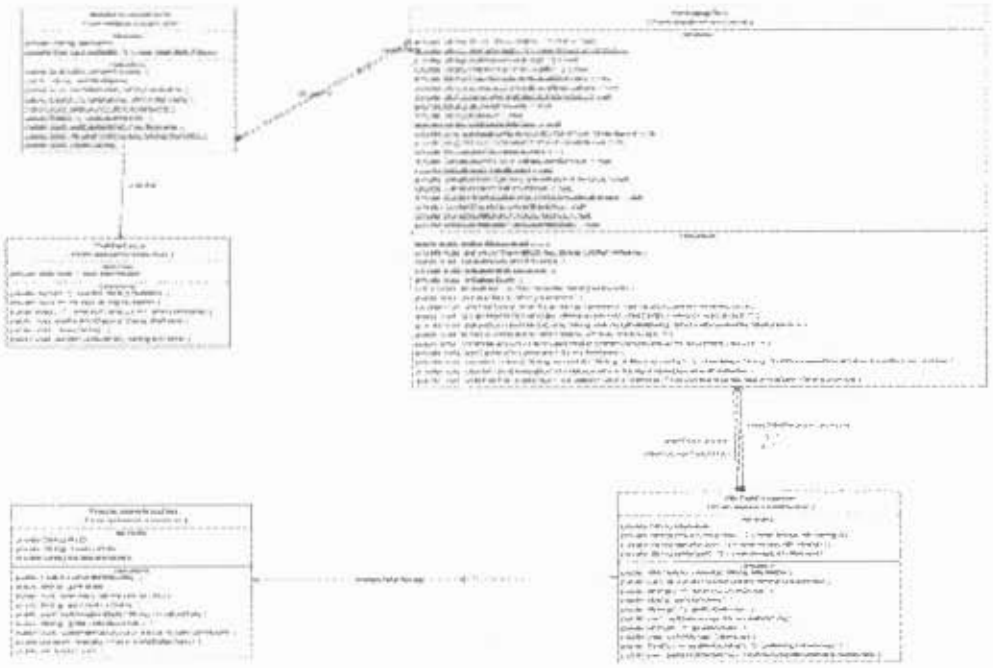
Run01	T01	Run02	T02	Run03	T03	Run04	T04	Run1	T1
1.19E+12	0.381117	1.19E+12	1.221933	1.19E+12	0.410967	1.19E+12	1.3246	1.19E+12	0.349133333
1.19E+12	0.812533	1.19E+12	1.38075	1.19E+12	0.564517	1.19E+12	1.469033	1.19E+12	1.5228
1.19E+12	1.996017	1.19E+12	1.648183	1.19E+12	0.676233	1.19E+12	1.579717	1.19E+12	1.637366667
1.19E+12	2.16305	1.19E+12	1.811683	1.19E+12	0.768217	1.19E+12	1.75325	1.19E+12	1.832466667
1.19E+12	2.292633	1.19E+12	2.4211	1.19E+12	0.876817	1.19E+12	1.872233	1.19E+12	2.005216667
1.19E+12	2.427667			1.19E+12	3.412017	1.19E+12	1.971217	1.19E+12	2.12395
1.19E+12	2.669333			1.19E+12	3.540367			1.19E+12	2.266833333
1.19E+12	2.942467			1.19E+12	3.652583			1.19E+12	2.489983333
1.19E+12	3.076983			1.19E+12	3.784567			1.19E+12	2.66585
	2.084644		1.69673		1.965143		1.661675		1.877066667

Run2	T2	Run3	T3	Run4	T4	Run5	T5	Run6	T6
1.19E+12	0.611767	1.19E+12	0.6167	1.19E+12	0.988967	1.19E+12	0.62085	1.19E+12	0.529167
1.19E+12	0.704	1.19E+12	0.7788	1.19E+12	1.127433	1.19E+12	0.73335	1.19E+12	0.6614
1.19E+12	0.814417	1.19E+12	0.93545	1.19E+12	1.24745	1.19E+12	0.85285	1.19E+12	0.897283
1.19E+12	0.926383	1.19E+12	1.045867	1.19E+12	1.374233	1.19E+12	0.944567	1.19E+12	1.014433
1.19E+12	1.054717	1.19E+12	1.17785	1.19E+12	1.5223	1.19E+12	1.073417	1.19E+12	1.11525
1.19E+12	1.166683	1.19E+12	3.7148	1.19E+12	1.650117	1.19E+12	1.190833	1.19E+12	1.20825
1.19E+12	1.273983	1.19E+12	3.83845	1.19E+12	1.767283	1.19E+12	3.933533	1.19E+12	3.1363
1.19E+12	1.387767	1.19E+12	3.973283	1.19E+12	1.9177	1.19E+12	4.032	1.19E+12	3.620517
1.19E+12	5.3392	1.19E+12	4.088383	1.19E+12	2.030717			1.19E+12	3.745733
	1.475435		2.241065		1.514022		1.672675		1.672675

Run7	T7	Run8	T8
1.19E+12	0.788933		
1.19E+12	0.965583	1.19E+12	0.650733
1.19E+12	1.07495	1.19E+12	0.741917
1.19E+12	1.2859	1.19E+12	1.0508
1.19E+12	1.6166	1.19E+12	1.351617
1.19E+12	1.742083	1.19E+12	1.46125
1.19E+12	1.87275	1.19E+12	1.627267
1.19E+12	2.00265	1.19E+12	1.732217
1.19E+12	2.943033	1.19E+12	1.826783
	1.588054		1.305323

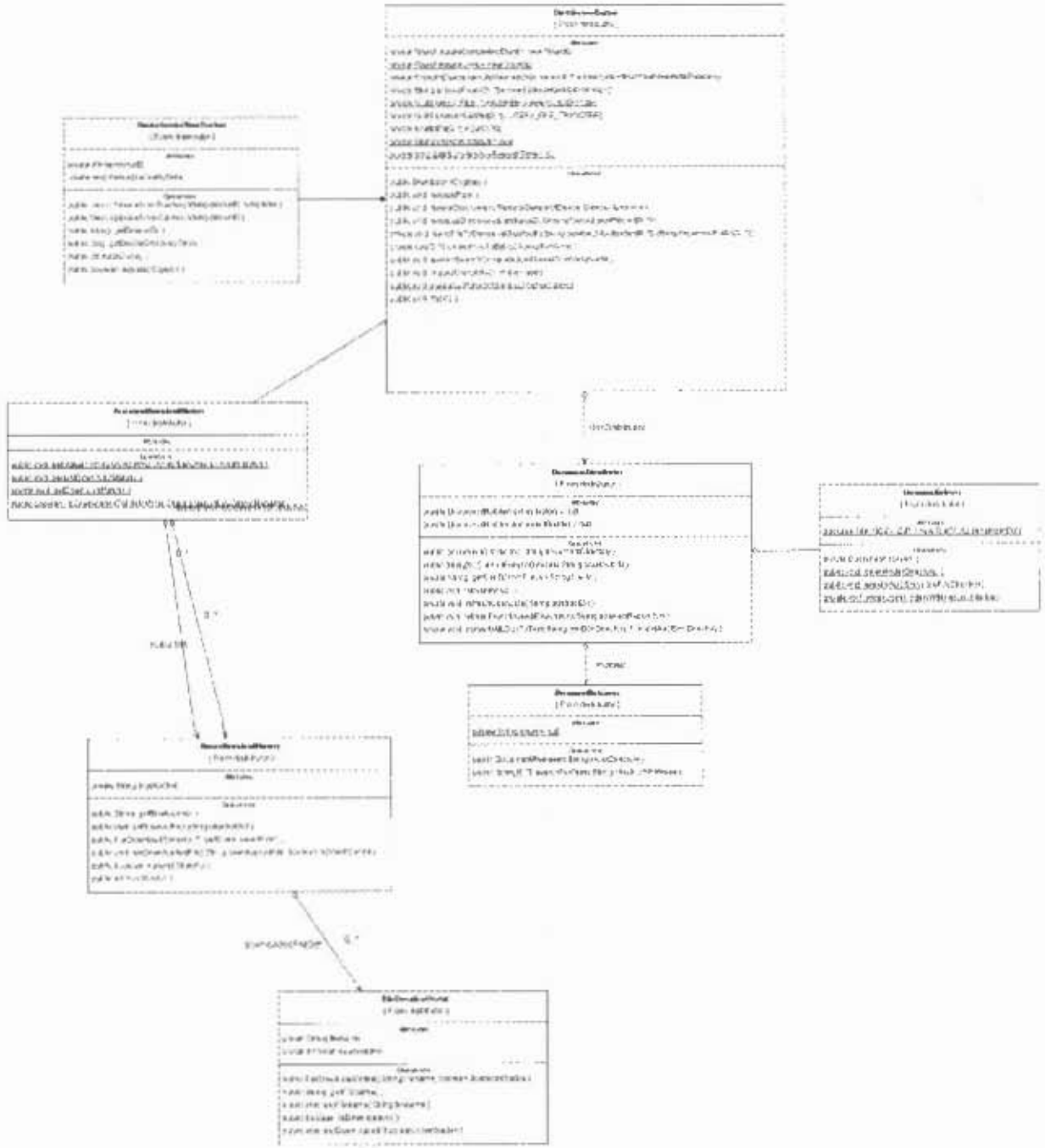
APPENDIX F

Detailed class diagram for Packaging module.



APPENDIX G

Detailed class diagram for distribution module.



APPENDIX H

A sample XML configuration file.

```
<?xml version="1.0" encoding="utf-8" ?>
- <!--This is the configuration content distribution system-->
<DistributeArchitecture>
<mobileAbstractsDir>C:\USERLISTtESTfOLDER</mobileAbstractsDir>
<!--This section is used for configuring of locations for various content
repositories/folders for the packaging module-->
<acquiredRepositoryContentDir>C:\ContentDistributionProject\VulaConten
t</acquiredRepositoryContentDir>
<userDeviceXMLRegistrationListDir>C:\ContentDistributionProject\Bluetoo
thAddrssXMLLists</userDeviceXMLRegistrationListDir>
<xmlAbstractsDir>C:\ContentDistributionProject\DocumentAbstracts</xml
AbstractsDir>
<indexDir>C:\LuceneIndexDir</indexDir>
- <!—time settings for the packaging and distribution module in seconds-->
<formattingModulePollingTime>30</formattingModulePollingTime>
<distributorModuleRefreshTime>10</distributorModuleRefreshTime>
<!--choose the sections of the document that you want to distribute to mobile devices→
<distributedSection>abstract</distributedSection>
<numberOfContentLines>12</numberOfContentLines>
</DistributeArchitecture>
```

APPENDIX I

Raw Data for number/size document download time

No Of Files	device1	device2	device3	Download time
1	1.297	1.219	1.265	1.26
2	2.51	2.51	2.557	2.53
3	3.762	3.465	3.699	3.64
4	4.972	4.891	4.909	4.92
5	6.009	6.148	6.195	6.12
6	7.237	7.426	7.488	7.38
7	8.625	8.578	8.39	8.53
8	9.663	9.663	9.928	9.75
9	10.865	11.318	10.834	11.01
10	11.879	12.348	12.301	12.18