Distributed Office Applications for Linux PDA

Stephen Asherson sasherso@cs.uct.ac.za
Trishan Valodia tvalodia@cs.uct.ac.za
Kutloisiso Mona kmona@cs.uct.ac.za
Ken MaGregor ken@cs.uct.ac.za

Abstract

Many countries within Africa suffer from lack of access to necessary technology resources; the resulting effect is a lack of computer literacy amongst these societies and in particular, learners within educational institutions. There have been many investigations regarding the use of Personal Digital Assistants (PDAs) within education; many of these investigations involve the use of productivity software suites such as Microsoft Pocket Office. Although these investigations have been undertaken, many of them involve proprietary software. There is a significant need for an Open Source Office suite designed for a PDA, which is able to serve as an alternative to proprietary software. This need is particularly high within the African context whereby financial resources are scarce. This project aims at exploring the possibility of providing an Open Source office applications suite for a PDA.

This project resulted in a word processor, spreadsheet application and presentation application ported to a Linux-based PDA. These cheaper devices are limited in their resources yet the resulting applications are not hindered by these limitations.

The applications rely on remotely accessing components that are present on a desktop machine, rather than accessing components locally on the PDA. This approach shifts the greater portion of processing, memory and storage requirements onto the desktop machine rather than to the limited PDA. The resulting application sizes are significantly smaller than the original complete applications.

As a result, educators and learners may make use of this software within a learning environment to share and gain computer literacy skills

Keywords: Spreadsheet, CORBA, Components, Open Source Software, Education, Word Processor, Presentation

Introduction

Handheld technology has recently experienced tremendous growth and advance, devices such as Cellular telephones and PDAs (Personal Digital Assistant) are now capable of providing sufficient computing power to perform a variety of tasks and support a number of software applications. The ability to host user written software on these handheld devices has lead to their application in a variety of different fields and environments; PDAs in particular have been involved in extensive research regarding their use in a variety of situations. One such area of particular importance to this project is the investigation regarding the use of PDA devices within the educational context.

Education is vital for the future of any country's welfare. The economy of any country relies heavily on the educational status of the generations entering the workplace. Several 3rd world countries situated in Africa are already trailing far behind other countries with respect to the educational skills present in their workforce. Owing to the fact that many of the economies and workforces worldwide are largely driven by technology, it is imperative that adequate computing education is incorporated into the educational system in any country. Unfortunately many of these 3rd world countries simply have a lack of access to such technology due largely to financial constraints and regional boundaries; a large factor for these high costs is the widespread use of proprietary software and the expensive hardware it requires.

One means of providing educational institutions and people in general better access to such technology is through the incorporation of Open Source software within education. The incorporation of Open Source software within these poorer countries, particularly within Africa, would allow people and developers in these countries to become part of a global Open Source group and essentially could provide a steep learning curve for learners in these countries. Open Source software will not only be beneficially through the learning aspect but also through the financial aspect. Open Source software is an ideal alternative to proprietary software without the high cost, Open Source software does not require the high licensing costs involved with proprietary software and is less dependant on expensive skills and hardware [2].

The aim of the project is to investigate the use of Open Source software together with PDAs in the educational context. Owing to the fact that Office suite productivity software (such as Microsoft Office) is extremely essential within a learning environment, the specific goal of the project is to provide Office suite functionality on a PDA solely through the use of Open Source software, this will provide institutions and users with a cheaper, more accessible alternative to proprietary applications such as Microsoft Office.

Design

Framework

A component framework is required that incorporates those components developed for this project as well as have an ability to incorporate future client and server components.

The component framework should provide solutions to the following requirements:

- Maintain a client application that installs and loads client-side components.
- Maintain a server application that allows developers to install components developed using the

Bonobo/ORBit component architecture.

- Dynamically searches and loads server components.
- Alerts the client application of all available server components when requested.
- Fulfils the request of a user by loading the selected application component.

Abiword

The word processor component used in the distributed environment will essentially provide basic word processing functionality. Owing to the fact that a PDA has limited resources, certain advanced functionality cannot be catered and is outside the scope of this project. The following are the basic features planned to be incorporated in the word processor component:

- File Operations: New file; Open file; and Save file.
- Edit Operations: Undo; Redo; Cut; Copy; Paste; and Select All.
- View Operations: Zoom functionality.
- Format Operations: Text Align; Bold; Italic; Underline; and Fonts.

The distributed environment will essentially consist of a single desktop machine running as a server and multiple PDAs running as clients. The clients and server are connected via a network protocol such as Ethernet or wireless. The following shows the process when a user wishes to load an application:

- The user runs the desired application on the PDA; this will load a thin-client application on the PDA.
- The client PDA application will then request a component corresponding to the user's application from the Server.
- The Server machine activates the component locally and delivers it to the client.
- The client then uses the component which appears to be running locally but is actually running on the server and requires little resources from the Client.

This distributed design is especially useful for use with PDAs, the client application is very small and few resources of the PDA are used. Even though the application is small, the use of a component technology such as Bonobo and CORBA would provide great functionality on these small PDAs without much strain on the PDA as most of the processing occurs on the server side. One of the main challenges of the system however is to achieve scalability whereby many users on handhelds or PDAs can use the system simultaneously.

MagicPoint

MagicPoint is an application that allows users view learning content using Linux machines. It is a lightweight, portable presentation viewer that allows slideshows to be viewed on any Linux machine. These slideshows may be written in any text editor using simple directives and tags for creating slides. It offers image display in various formats, background images and template suppot. It also has the ability to export slideshow to an html format which can be viewed in most popular browsers.

To use MagicPoint for use on the Linux PDA, it would have to be ported and cross-compiled for the ARM architecture. Once implemented the PDA would allow native creation of MagicPoint presentations using VI. These presentations may be viewed on the Familiar Linux PDA without the need for additional libraries.

YaSP

YaSP is a distributed application i.e. there is part of the application is a client and the other part is a server. The server is a CORBA component and the client calls the services implemented by the server. Figure 1 below shows the architecture overview of YaSP. The client calls methods or services provided by the server and the server performs the service and sends the response back to the client. All communication is done via the CORBA ORB.

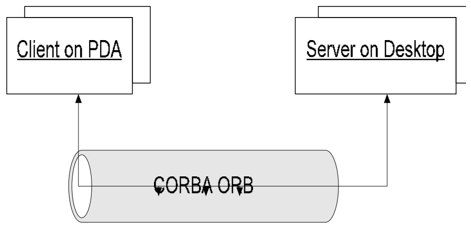


Figure 1 Architecture Overview

Only file operations were implemented in this phase. Figure 2 depicts the use case scenarios of YaSP.

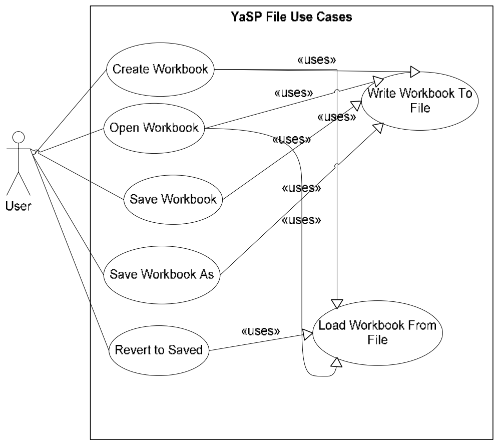


Figure 2 YaSP Use Case Diagram

Implementation

Framework

This section will discuss the implementation of a component framework that will incorporate those components as well as have to ability to incorporate future client - server components and applications.

The following diagram illustrates the relationship between the client and server application of the framework:

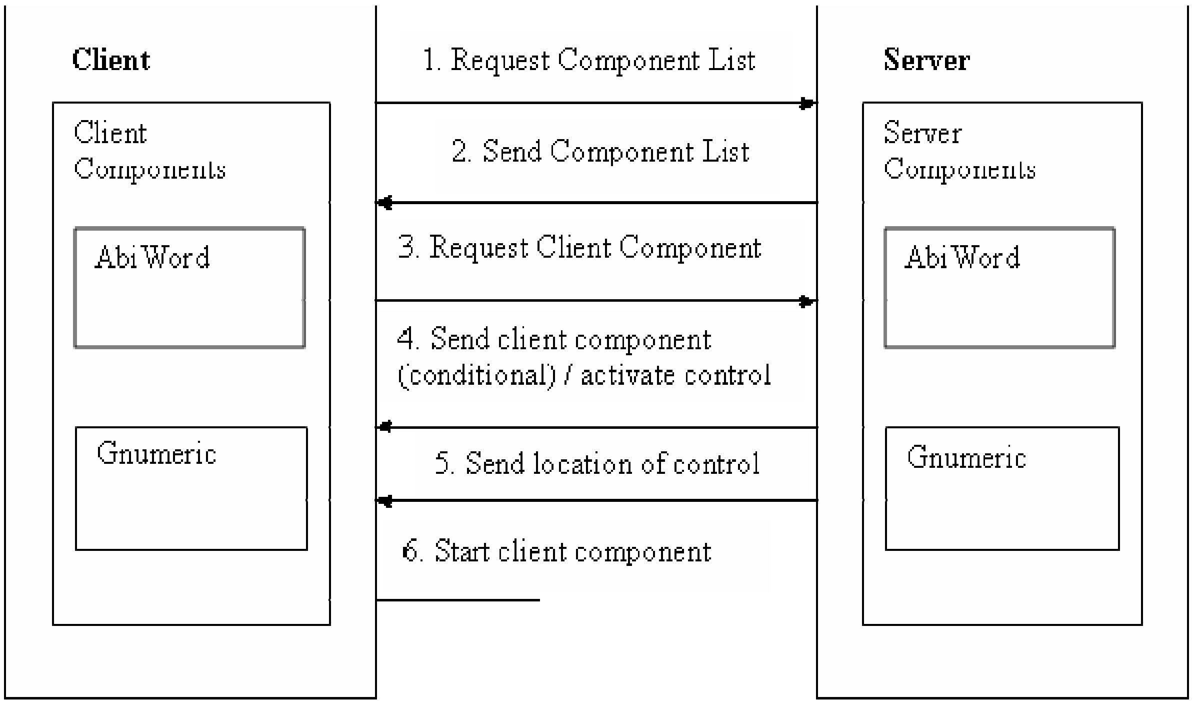

Client
Client Components
Abi Word
Gnumeric
1. Request Component List
2. Send Component List
3. Request Client Component
4. Send client component (conditional) / activate control
5. Send location of control
6. Start client component
Server
Server Components
Abi Word
Gnumeric

Figure 3 Framework Client and Server Relationship

When the server starts, it traverses the sub-directories searching for available server components. Server components are identified by a component descriptor within its sub-directory. The descriptor file is a text file containing the name of the application component, the name of the object reference filename, the server component executable filename and the client component executable filename. The name of the descriptor file is "component.descriptor". An example of a component descriptor file:

Abiword

Abiword.ref

AbiControlServer

AbiControlClient

When requested, the server sends the client-side details of the components to the client PDA.

Figure 4 below is a screenshot showing the client display after a list of components has been requested.

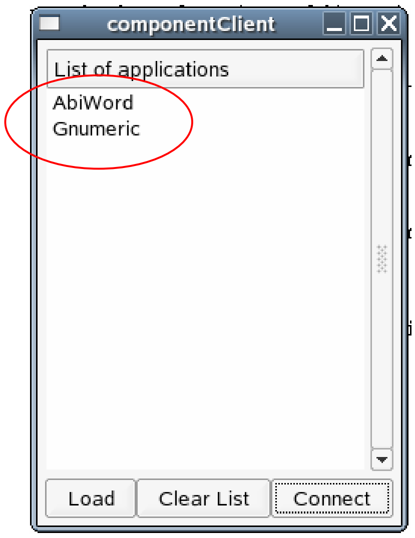


Figure 4 A connection has been made

The user may select a component to load by first selecting a component from the list and then clicking the "load" button.

The significance of the framework lies in the storage requirement of the client components. The AbiWord client component uses 0.5 MB of storage on the PDA, while the entire AbiWord source code amounts to approximately 30 MB.

AbiWord

The implementation of the word processor component was achieved using a word processor application known as AbiWord. Owing to the fact that distributed computing will be used to provide the word processing functionality through component technologies, AbiWord is an ideal word processor to use as it already provides support for the Bonobo/CORBA component technologies.

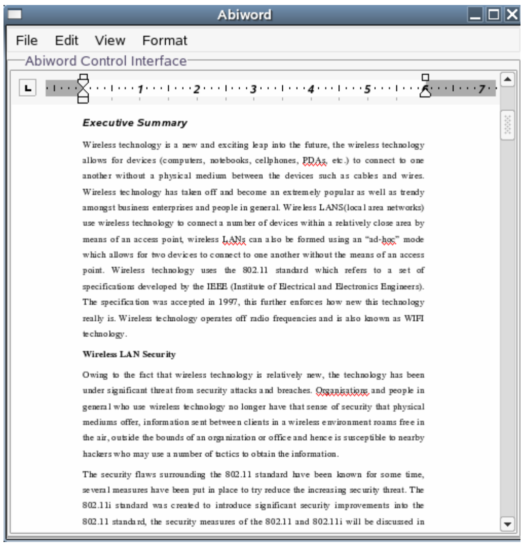
Bonobo and CORBA are component technologies which allow different applications running on different architectures and operating systems to communicate via components. Components can be thought of as individual pieces of software and are specified through strict IDL (Interface Definition Langauge) interfaces; these interfaces are used to allow absolute communicate between client and server irrespective of programming language used [1].

The AbiWord word processor provides a Bonobo component which can be used as a control within a user interface; this essentially means an AbiWord Bonobo control can be obtained remotely from a server and embedded within a graphical user interface on the client side, the word processing functionality of the control can then be used by the client via the user interface.

When a client user wants to load the AbiWord application, the user will start the framework and select the AbiWord component from the list. The client application will then request the activation of the AbiWord component from the Server. The Server will then activate the AbiWord component locally using the Bonobo activation framework as follows:

```
Bonobo_Control AbiControl =
bonobo_get_object
("OAFIID:GNOME_AbiWord_Control",
bonobo/Control", &ev);
```

Once the server has activated the component, the server obtains the CORBA ID of the object and writes the ID to a file on disk. The file is then sent to the client whereby the client application uses the CORBA ID from the file to obtain a reference to the AbiWord component residing on the server. The component is then embedded inside a graphical user interface on the client side and this enable the user to manipulate documents via communications with the AbiWord Bonobo component.



Executive Summary

Wireless technology is a new and exciting leap into the future, the wireless technology allows for devices (computers, notebooks, cellphones, PDAs, etc.) to connect to one another without a physical medium between the devices such as cables and wires. Wireless technology has taken off and become an extremely popular as well as trendy amongst business enterprises and people in general. Wireless LANS(local area networks) use wireless technology to connect a number of devices within a relatively close area by means of an access point, wireless LANs can also be formed using an "ad-hoc" mode which allows for two devices to connect to one another without the means of an access point. Wireless technology uses the 802.11 standard which refers to a set of specifications developed by the IEEE (Institute of Electrical and Electronics Engineers). The specification was accepted in 1997, this further enforces how new this technology really is. Wireless technology operates off radio frequencies and is also known as WIFI technology.

Wireless LAN Security

Owing to the fact that wireless technology is relatively new, the technology has been under significant threat from security attacks and breaches. Organisations and people in general who use wireless technology no longer have that sense of security that physical mediums offer, information sent between clients in a wireless environment roams free in the air, outside the bounds of an organization or office and hence is susceptible to nearby hackers who may use a number of tactics to obtain the information.

The security flaws surrounding the 802.11 standard have been known for some time, several measures have been put in place to try reduce the increasing security threat. The 802.11i standard was created to introduce significant security improvements into the 802.11 standard, the security measures of the 802.11 and 802.11i will be discussed in

Figure 5 The Abiword Component Interface

MagicPoint

MagicPoint was the only application that successfully cross-compiled to the Linux PDA.

The other packages were not successfully compiled for the following reasons:

- Missing dependency libraries
- Missing compilation tools
- Incompatible compilation tools.

Compiling MagicPoint on the PDA required editing the source makefiles. The makefiles compiled the source code using gcc. One of the switches used during compilation were “-m32 -c -O2 -march=i586 -mcpu=i686”. These switches did not allow a successful compilation of MagicPoint within the Scratchbox and had to be removed from the compilation process.

The MagicPoint application makes use of the entire graphical area during a slideshow. The slide show may be controlled using the arrow keys of the PDA as well as by making use of the stylus.

Slideshows for MagicPoint may be created directly on the PDA using the native VI or may be created on a desktop machine with a standard text editor and copied over to the PDA.

Having successfully compiled MagicPoint for the PDA gives the educator a chance to provide course content, in the form of a slideshow to the learners.

YaSP

Initially we attempted to implement Yet another Spreadsheet Program (YaSP) using Openoffice.org office suite. This failed for a number of reasons but the main one was that UNO Runtime Environment required Openoffice.org and Java to be installed on the machine.

Gnumeric was used as the CORBA server that was to be called by our client. The latest version of Gnumeric was 1.5 however we used version 1.2.13. This earlier version of Gnumeric was used because it was the latest version of Gnumeric that still had Bonobo support. Since this version Bonobo support in Gnumeric has been discontinued.

Orbit and Bonobo were used as the CORBA ORB implementation. Bonobo has an object activation framework (OAF). Components register themselves in OAF by installing an XML description file (`.serve`) with their location information (i.e., where to find the executable or shared library), and arbitrary other information. The framework works very well for clients on the same machine as the server however we were unable to query for servers using remote clients.

In order to request a service from a server object the ORB has to locate a server object at run-time and the client application requires a reference to it. This reference is called an Interoperable Object Reference (IOR). An IOR is a text string encoded in a specific way, such that a client ORB can decode the IOR to locate the remote server object. It contains enough information to allow:

- A request to be directed to the correct server (host, port number)
- An object to be located or created (class name, instance data)

This file is transferred from the machine hosting Gnumeric to the PDA using trivial transfer protocol (TFTP) program. This program was also used to transfer other files.

When the client is started, i.e. the YaSP program on the PDA it executes the command:

```
xhost +[nameOfServerMachine]
```

This command is needed in order to allow the control to be displayed on the PDA. Otherwise there will be authentication error since the server will not be allowed to use the PDA display.

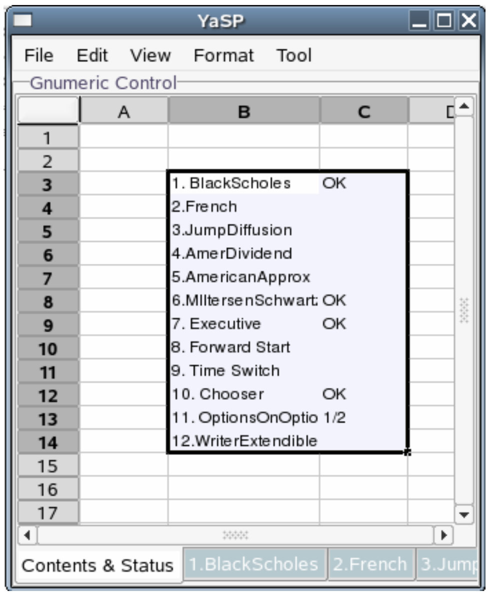
Because OAF could not activate remote components we created our own server. The server activates Gnumeric component using the OAF and creates a file containing its IOR string. The TFTP program then copied the IOR string file from the server machine to the PDA. Normally a Bonobo component is activated using the code:

```
Control = bonobo_get_object
("OAFIID:GNOME_Control","Bonobo/Con
trol", &ev);
```

The string `"OAFIID:GNOME_Control"` is `identification string of the object`. However for Gnumeric this does not work. We thus used the following code in the server to activate Gnumeric:

```
control = bonobo_get_object("file:/
test.gnm","IDL:Bonobo/Control:1.0",
&ev);
```

This queries the OAF for a component that can handle the given file. When the component was obtained it was exported to file as an IOR-string. TFTP is then used to copy the IOR file to the PDA. The client on the PDA then creates a Gnumeric object reference using the IOR. The object reference is then used to get an interface that has been implemented by Gnumeric. The interface is used to call all the functions of Gnumeric from the client on the PDA. Figure 6 below shows the screenshot of YaSP.



	A	B	C
1			
2			
3		1. BlackScholes	OK
4		2.French	
5		3.JumpDiffusion	
6		4.AmerDividend	
7		5.AmericanApprox	
8		6.MIltersenSchwart:	OK
9		7. Executive	OK
10		8. Forward Start	
11		9. Time Switch	
12		10. Chooser	OK
13		11. OptionsOnOptio	1/2
14		12.WriterExtendible	
15			
16			
17			

Figure 6 Screenshot of YaSP

Unfortunately only persistence and zoomable interfaces are implemented in Gnumeric and thus the only functionality besides viewing that could be implemented was zooming in and out.

Conclusion

The project resulted in a client-server application that allows users to remotely load components over a network. The framework application created allows developers to add their applications to the framework providing PDA users with the functionality provided. The applications designed for the PDA may be component based or standalone applications.

Although the Initial aim to develop a stand-alone application to run on the PDA was abandoned, the distributed environment which followed for the AbiWord component provided the use of basic word processing functionality on a PDA.

Gnumeric is ineffective as a component. None of the interfaces that are required to implement features of a usable spreadsheet are not implemented. Due to this limitation YaSP cannot be implemented using Gnumeric as the server spreadsheet.

However the use of such Open Source software is a stepping stone in providing less fortunate users, particularly learners with productivity software other than proprietary software. There is great potential in both the use of Open Source software and handheld devices in within the educational context; this was definitely evident throughout the duration of the project.

Much was learnt from using the Bonobo and CORBA component technologies to enable communication between the clients and the server, the distributed environment was ideal for the PDAs due to the limited resources available on the PDA; there is also great potential for the use of a distributed environment with component technologies as this enables older, cheaper hardware to provide new functionality though components which would previously have been impossible.

MagicPoint resulted in a contribution to the overall office suite in the form of a portable presentation application. The MagicPoint port is not a distributed application and therfore files may not be stored directly onto a server for backup purposes while the resulting binary is slightly larger than the AbiWord and Gnumeric client components. However, users may make presentations and view them without the need of a component server. This makes the application more portable while network connectivity is not required to load presentations.

As a future works, an investigation regarding the use of distributed components using the UNO technology provided by the OpenOffice.org office suite could be undertaken. This investigation would be interesting and possibly provide further functionality than the AbiWord and Gnumeric components.

References

1. CORBA FAQ. The Object Management Group. Available at http://www.omg.org/gettingstarted/corbafaq.htm
2. Victor van Reijswoud, Corrado Topi, Alternative routes in the digital world – Open Source software in Africa. Open Source research community. Available at http://opensource.mit.edu/papers/reijswoudtopi.pdf
3. O'Reilly T. (1999) Lessons from open-source software development, *Communications of the ACM*, 42(4), 32-37.
4. Feller, J. & Fitzgerald, B. (2000) A framework analysis of the open source software development paradigm, *Proceedings of the 21st International Conference on Information Systems*, Brisbane, Australia. ACM Press. 58-69.
5. Guy Antony Halse and Alfredo Terzoli. Open Source in South African Schools: Two Case Studies. Centre of Excellence in Distributed Multimedia, Rhodes University. Available: http://eprints.ru.ac.za/100/01/HALSE-Highway-Africa-2002.pdf Date accessed: 01 October 2005
6. SchoolNet South Africa. http://www.school.za/