ITB Journal

Issue Number 6, Dec 2002

Contents

Papers for submission to the next ITB Journal should be sent to the editor at the address below. Alternatively, papers can be submitted in MS-Word format via email to brian.nolan@itb.ie

Brian Nolan
Editor
ITB Journal
Institute of Technology Blanchardstown
Blanchardstown Road North
Blanchardstown
Dublin 15

Editorial

I am delighted to introduce the fifth edition of the ITB Journal, the academic journal of the Institute of Technology Blanchardstown. The aim and purpose of the journal is to provide a forum whereby the members of ITB, visitors and guest contributors from other third level colleges can publish an article on their research in a multidisciplinary journal. The hope is that by offering the chance to bring their work out of their specialised area into a wider forum, they will share their work with the broader community at ITB and other academic institutions.

In this issue, we have papes on a wide range of topics. In their paper on Virtual Credit Card Processing System, G. Gray, K. Church, and T. Ayres, of ITB explore virtual techniques for safeguarding online financial transactions. The JAM Suite paper from B. Nolan, M. Cummins, I. Flanagan, and N. O'Brien of ITB discusses the development in Java of software that delivers a voice-enabled network-based virtual band application. GlobalCom from D. Barber, C. Gildea, and G. Byrne of ITB shows how an enterprise can communicate with its employees through the use of a unified messaging system. In Learning Through Dialogue (LTD) M. Smith, J. Cook and M.Oliver of ITB, London Metropolitan University, and University College London discuss a toolkit to support the process of planning for effective use of dialogue in learning. The paper from S. Sheridan of ITB provides a Java framework for using and processing computer vision. In their paper Longer Than A Telephone Wire, C. Reynolds, M. Smith, and M. Woodman of Middlesex University and ITB discuss how the new technology of voice firewalls can be used to counter ubiquitous lie detection. The deployment of an XML framework for structured data exchange between medical devices is the topic of the paper from U. Neuhaus, P.Walsh. and K. Wente from Cork Institute of Technology and Fachhochschule Darmstadt. B. Nolan from ITB discusses issues relating to Lexical Semantics and Patterns of Causation in his paper. G. Gray, K. O'Connor of ITB examine the technology infrastructure required for the deployment of Web Services. Extending physical simulation to the audio domain for enhanced realism in game engines is the subject of the paper by G. McCann, and H. McCabe of ITB. R. Gallery, and D. M. Shakya of ITB discuss hardware and software codesign for multimedia capable portable devices using SystemC in their paper. D. Barber and A. Keane of ITB discuss issue relating to Unified Messaging Systems: An Evolutionary Overview. The paper from Karen Church of ITB, "Searching ……", won the Higher Education and Training Awards Council Prize for Computing in December 2002.

We hope that you enjoy the papers in this issue of the ITB Journal.

Brian Nolan
Editor
ITB Journal
Institute of Technology Blanchardstown
Blanchardstown Road North
Blanchardstown
Dublin 15

Virtual Credit Card Processing System

Geraldine Gray, Karen Church, Tony Ayres

Institute of Technology, Blanchardstown, Dublin 15, Ireland

Abstract:

The virtual credit card processing system is an e-business system we have developed which provides a secure and universal mechanism for making purchases over the Internet. The system uses Remote Method Invocation (RMI), Java Server Pages (JSP), Java Servlets and Java Database Connectivity (JDBC). We also look at the possibility of implementing the system using the Web Services architecture.

1. Introduction:

The Virtual Credit Card Processing System is a web-based multi-tiered distributed application. It is be powered by Servlet/JSP and RMI technology and implements an Oracle database and MySQL database for back-end processing. The system is divided into three separate components, namely an online shop, an on-line bank, and the virtual credit card provider. The online shop comprises of a fully functional shopping cart through which users can purchase goods. The online bank consists of a number of different utilities the user can use to maintain their banking information. The virtual credit card provider generates unique virtual credit card numbers which it transmits to banks using a combination of RMI and Encryption. Users request a virtual credit card number from the online bank. The bank then requests a credit card from the Virtual Credit Card Provider. Users then use this credit card in online shops to make purchases. Shops validate the credit card number used with the bank.

The banking system enables the following operations to be performed: Open/close accounts, Request virtual credit card numbers, Check account balance/details, Change pin number, Transfer money, Pay bills, Use direct debiting functions. The shopping cart system enables the following operations to be performed: Find products, View Products, Buy Products, Register, Login, Post Comments. The virtual credit card component carries out the following operations: Creating the unique credit card numbers, Updating and maintaining transaction logs associated with these credit card numbers, encrypting the credit card numbers and transmitting them using RMI for use by the user.

2. Details of Java Technologies used

2.1 Distributed Technology

Each of the three components runs on its own hardware platform at a remote site. Implementing this distributed architecture requires middleware to provide a programming abstraction between applications running on separate machines We are going to use Java's Remote Method Invocation (RMI) package to provide this programming abstraction. [13,14]

Java RMI is a distributed object model for the Java Platform. It allows communication between objects running on different hosts through a series of message passing. Object methods can be referenced between different computers across a network, and real objects can be passed as arguments and return values during method invocation. Java RMI uses object serialization to convert object graphs to byte-streams for transport. Any Java object type can be passed during invocation, including primitive types, core classes and user-defined classes. The ability of a system to pass actual objects enables clean system design [1,2].

Java RMI provides a fully object-oriented distributed environment. RMI also operates naturally in the Java area, which allows developers to program within a single object model, the Java model. This removes a great deal of complexity. RMI requires no mapping to common interface definition languages. The syntax of remote method invocations is almost identical to local method invocations [1].

To use RMI successfully, you need to create a client, a server, an interface and an implementation of that interface. The interface is simply a list of all the methods accessible remotely by the client. The implementation is the mechanism for implementing the methods listed in the interface. When these objects are created, the server and the implementation run on one machine while the client runs on a separate machine. [2].

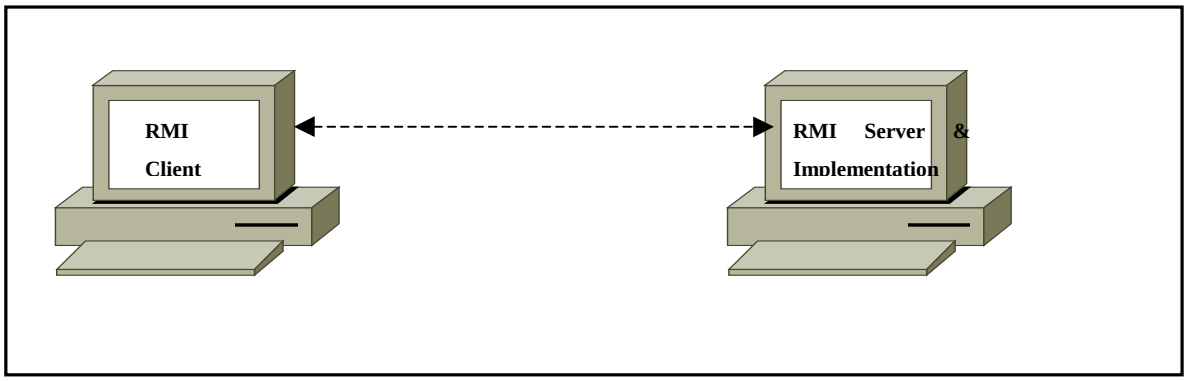


Figure 1: RMI architecture

2.2 Banking System

The banking system comprises of three main components. The first is a web-based application that represents the bank. This application provides the interface with which the user interacts. The second component is a communications medium between the bank and the virtual credit card component; the third component of the bank is a communications medium between the bank and the online shop.

The first requirement of the banking system is a web-based application that encompasses all of the functionality provided by the bank to the user. The user interface was developed in HTML, with the processing implemented using Java Servlets, connecting to an Oracle database using JDBC.

Java Servlets allow the application to be dynamic in nature since the output presented to the user depends on the inputs and requests made by the user.[7] Java Servlets are a Java Based Development tool that enable the generation of dynamic web pages using a request, response mechanism. In this HTTP based request-response paradigm, a client user agent (a web-based interface viewed through a web browser) establishes a connection with a web server and sends a request to the server. The server then issues a response. For web application development, the servlet API provides three primary abstractions: HTTP requests, request processing based on some application logic, and HTTP responses. In addition, the servlet API also provides a mechanism for session tracking and state management.[1] Being a Java technology, Java Servlets inherit all of the advantages that the Java language provides. This includes portability, scalability, robustness and simplicity. Each Servlet created processes a request made by users and then presents a response to the user in HTML format.

The majority of the Servlets we created require a connection to a database holding each users banking information. This connection was implemented using Java Database Connectivity (JDBC). Figure 2 below illustrates the role of Java Servlets and JDBC in this system. As can be seen from the diagram, the banking system is implemented in a three-tier architecture, comprising of a client, a server and a database.

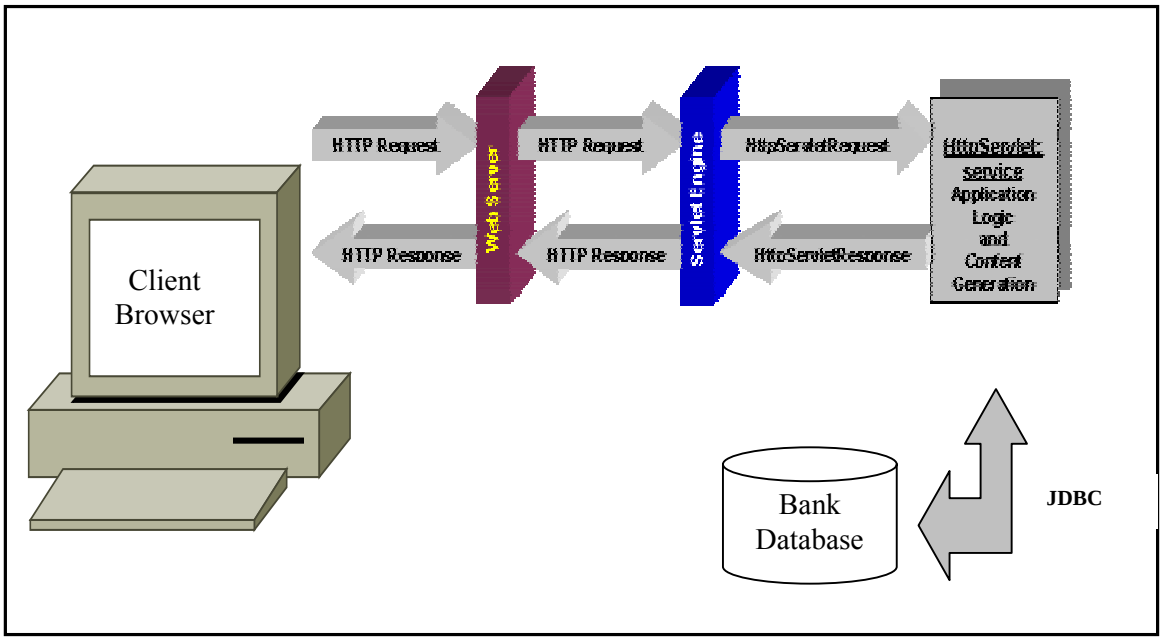


Figure 2: Banking system architecture

The second requirement of the banking system is an RMI component that communicates with the Virtual Credit Card component. The Virtual Credit Card component holds the RMI server and the implementation. The banking system holds the corresponding RMI client to access the methods provided by the RMI Server. The RMI client requests a virtual credit card number for use by the user. The RMI client is in fact a Servlet. This meant importing the java RMI package to facilitate binding the client to the server, i.e. inform the client of the location of the server so that communication can conduct successfully between these objects.

The third requirement of the banking system is a second RMI component. This RMI component allows communication to take place between the online shop and the banking system. When a user purchases goods using a virtual credit card in the online shop, this credit card number is sent back to the bank in a secure manner, so that the users account can be validated and debited for the amount of the purchase made. In this case the bank needs to hold an RMI server and an implementation. The server's primary function is to take the credit card number received from the shop and to carry out some processing and validation on the users account. This server is implemented as a servlet. The Java Servlet package

provides a RemoteHttpServlet class which allows any HTTP servlet to act as an RMI server. This package enables all RMI server operations to be carried out including binding itself to the RMI registry, logging any errors and providing remote methods accessible by the client.[1,7]

Figure 3 below illustrates the programming modules for the banking system:

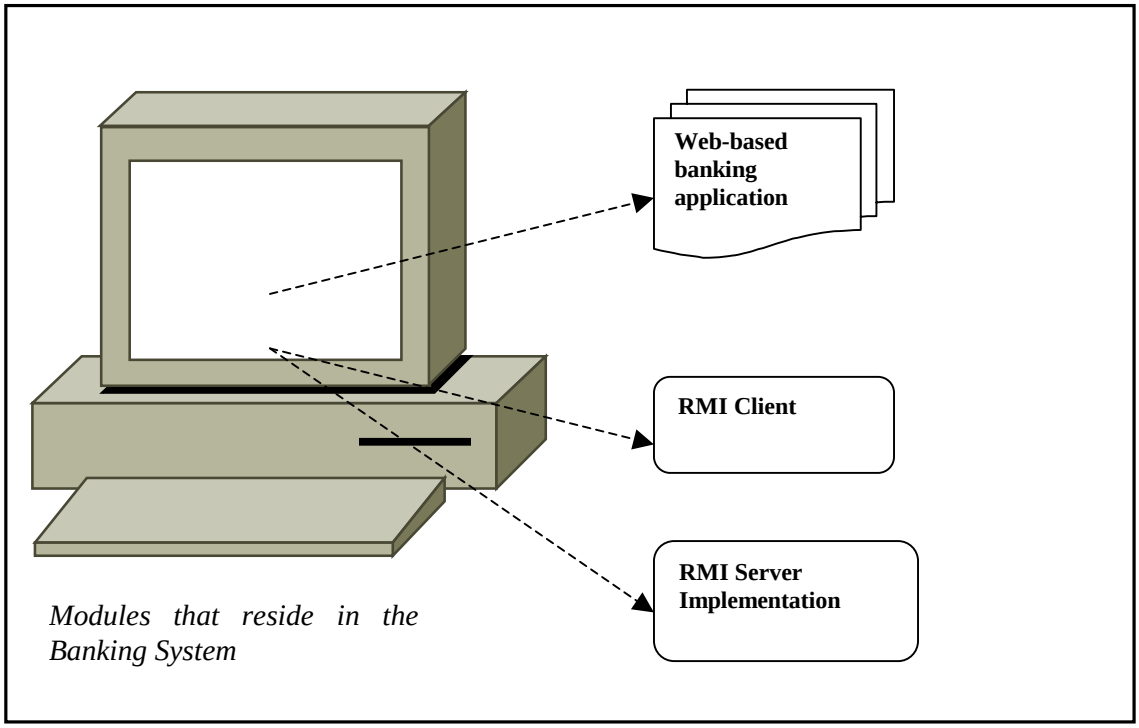


Figure 3: Banking system modules

2.3 Online Shop

The online shop is built using Java Server Pages and MYSQL database. Java Server Pages allow segments of Java code to be included in regular HTML documents. The Java code can be evaluated and the result printed out on the HTML page. Essentially the JSP will be compiled into a Java Servlet by the JSP engine. JSP offers an advantage over Servlets, in that it combines the power of the Java programming language with the simplicity of HTML. [1]

JSP has access to the majority of the API's in the Java language. As with Servlets, JSP also has methods for dealing with HTTP Requests, session tracking and HTTP Responses. Since the Servlet and JSP architecture are implemented and activated differently, JSP has request, response and session objects available to it at all times. This is particularly useful for an

application such as an online shop, as functionality such as tracking items in a shopping cart or identifying if a user is still logged in can be easily employed.

We selected Caucho's Resin 2.0.5 JSP/Servlet engine for the purpose of building the online shop. [15] When dealing with Servlet and JSP technology, Tomcat, the official Sun implementation of the JSP/Servlet specification is often the first Servlet engine that comes to mind. However, we found Tomcat to be slow and unreliable and for this reason we chose Resin.

MYSQL is a popular open source database, widely used for web based applications. Dynamic web applications with MYSQL traditionally used PHP to communicate with the database. However MYSQL-JDBC drivers exist to allow Java application's, or in this instance Java Server Pages, to communicate with MYSQL databases. With this set-up, we have an efficient combination of open source technology and platform independent Java technology. The MYSQL driver we used in this project was the mm.mysql driver, an open source JDBC driver. [16]

All product information on goods for sale in the shop is held in a database; JSP is responsible for retrieving the information based on customer's request. A database connection bean was used to connect to the database. This bean handles connections to the database, executing queries, returning result set objects and closing connections. Session tracking will be used to check the items in the users cart and to test whether or not the user is logged in.

Remote Method Invocation is used at the purchase stage of the shop. The user provides a virtual credit card number, the system will take this number and use RMI to call a method on the bank server to validate that the number is authentic. Based on the response of the bank object, the system will output the success or failure of the purchase attempt. The RMI object will be called from a Java Servlet. This servlet is also used to fulfil the users order, the users purchase information is stored in the ordering table of the database.

2.4 Virtual Credit Card Component

The Virtual Credit Card Component of the system comprises of two separate Java objects. The first java object is responsible for generating random credit card numbers, the second

java object is responsible for communicating with the credit card pool and the banking system in order to process requests from users of the system.

The first requirement of the virtual credit card component is the generation of a pool of unique credit card numbers. The Java API specification includes a java.util.Random package, which allows programmers to generate random numbers. This class called "Random" uses a 48-bit seed, which is modified using a linear formula. When you create a new random number generator with this package, its seed is initialised to a value based on the current time. To ensure that each credit card number is unique, we are going to use the java.util.Date package.[7] The Date package provides a getDate() and a getTime() method, which returns the current date and time in the following format:

- Day, Month, Year, Hour, Minute, Second, Millisecond, etc

As is known, time is constantly changing, so each time we call the getDate() and getTime() methods the numbers returned will be different from the previous. If we take the last 8 digits from the number returned we are guaranteed that they will be different from the previous numbers. This provided us with the unique element we required. To generate the 16-digit credit card number we require we use the java.util.Random package to generate an 8 digit random number which will serve as the first 8 digits of the credit card number. We can then bind these 8 random digits with the 8 unique digits we generated from the getDate(), getTime() methods. An example is shown below, followed by java code snippets to illustrate this process fully: Today's microprocessors can calculate vast quantities of data in nano-seconds. In testing the getDate() methods, we found that the processor ran so fast, that identical dates were returned. We introduced a thread to pause the execution of the getDate method so that a unique number was returned. To guarantee uniqueness, we used the java.util.Random package to generate a subsequent seed which was appended to the number returned from the date method. This ensured a completely random and unique credit card number.

2343 4566 – generated with java.util.Random
1223 5634 – generated with java.util.Date
2343 4566 1223 5634 – resulting credit card number

```
//random component
Random random=new Random(16);
long rand = random.nextLong()*random.nextLong();

//unique component – to be contained in a for loop

java.util.Date d = new java.util.Date();
long mili = d.getTime();
Long longmili = new Long(mili + rand);
String num = longmili.toString();
card_numbers[i] = num;
```

The second objective of the virtual credit card component of the system is to provide a communications medium between the pool of generated credit card numbers and the banking system. This is done using RMI as discussed in section 2.2. above. A series of requests will be carried out by users of the system, which are sent to the virtual credit card component for processing. These requests include requesting a credit card number from the pool and returning the credit card number to the pool.

The deployment diagram in Figure 4 below shows how the 3 systems interact using RMI. The shop system acts as an RMI client for the bank systems RMI Server and the bank system is a client of the Virtual Credit Card RMI server.

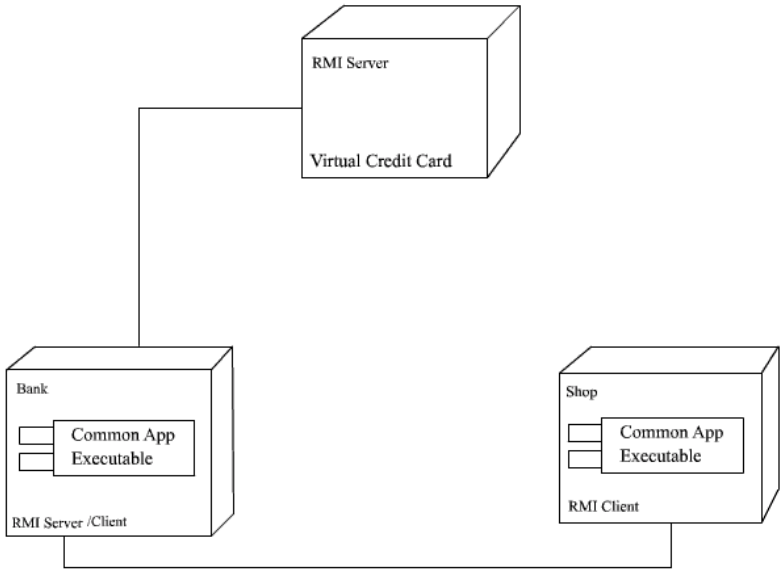


Figure 4: Deployment diagram

3. Future Considerations:

All the technologies as discussed in section 2 above were implemented successfully. However, an inevitable trait of technology is its tendency to change and evolve at a very fast pace. When designing the Virtual Credit Card Processing System using Java as the main programming language, technologies like RMI where considered first and foremost to cater for the distributed nature of the system. Since that time more technologies have emerged that could now be used to cater for this distributed system. These include protocols and languages like WSDL (Web Services Description Language)[10], SOAP (Simple Object Access Protocol)[5,6] and UDDI (Universal Description, Discovery, and Integration)[4] which together can be used to develop platform and language independent distributed web applications.

A web service is a service, available over a network, of components which can be called from a remote application. A WSDL file describes the Web service, enabling service providers and requesting clients to communicate with each other. WSDL defines XML syntax for describing network services as collections of communication endpoints capable of exchanging messages. Documentation for distributed applications and detailing the processes involved in communication between these applications is provided by WSDL service definitions. WSDL allows endpoints and their messages to communicate regardless of the message formats or network protocols being used. Primary uses for WSDL is the design of specifications to invoke and operate Web services on the Internet and to access and invoke remote applications and databases [9].

Simple Object Access Protocol, or SOAP is a protocol for exchanging information in a decentralized, distributed environment. It defines a mechanism for passing information and parameters between clients and servers. An advantage of SOAP is that it is a platform, object and programming language independent protocol and so can be used to create open and compatible systems. This protocol is XML-based and consists of three parts. An *envelope* that defines a structure for describing the contents of a message and how to process it, a set of *encoding rules* for expressing instances of application-defined data types, and a *method* for representing remote procedure calls and responses. A particular advantage of SOAP over a protocol like Java Remote Method Protocol (JRMP) for Java Remote Method Invocation (RMI) is that SOAP uses XML and is therefore text-based. XML is easier to read than a binary stream, making SOAP-based applications easier to debug [5,8].

Universal Description, Discovery, and Integration (UDDI) is like a central point for Web services. A UDDI repository stores descriptions about companies and the services they offer in a common XML format so it is like an information registry of Web services. This service is used by web service brokers to register services that service requesters can then discover and invoke. Web-based applications interact with a UDDI repository using SOAP message. [4,11,12].

For this project, the Virtual Credit Card provider could offer virtual credit cards as a web service. The component could be developed using Enterprise Java Beans. A WSDL XML file would provide information about the services being offered by the EJB's. Banks could then develop their own applications to use this web service.

4. References:

[1] Professional Java Server Programming: with Servlets, JavaServer Pages (JSP), XML, Enterprise JavaBeans (EJB), JNDI, CORBA, Jini and Javaspaces by Andrew Patzer, et al, Wrox Press, 1-861002-77-7, 2000
[2] Distributed Systems, Concepts and Designs, 3rd Edition, Coulouris et al, Addison Wesley, 0-210-61918-0, 2001
[3] Enterprise JavaBeans (3rd Edition), by Richard Monson-Haefel, O Reilly, 0-596-002268, 2001
[4] www.uddi.org/
[5] www.w3.org/TR/SOAP/
[6] www.intelligententerprise.com/010327/feat2_1.shtml
[7] http://www.java.sun.com
[8] http://www.onjava.com/
[9] www.w3.org/TR/wsdl
[10] http://www.capeclear.com/products/capeconnect/terms.shtml
[11] http://uddi.microsoft.com/default.aspx
[12] www.ibm.com/services/uddi/
[13] http://java.sun.com/products/jdk/rmi/
[14] http://developer.java.sun.com/developer/onlineTraining/rmi/RMI.html
[15] www.caucho.com
[16] www.mysql.com

The JAM Suite - A Voice-enabled Network-based Virtual Band Application

Brian Nolan, Martin Cummins, Irene Flanagan, Niall O'Brien

Institute of Technology Blanchardstown

Abstract

The Java Audio Music-suite (J.A.M. for short) is a suite of applications that provides tools and audio processing utilities for musicians. It's core functionality includes a means whereby musicians who are geographically dispersed can play music together. An additional utility determines the actual musical notes from a piece of music using Fourier Transform techniques. All of these functions are complemented by a voice-enabled, animated help agent that takes in voice commands using speech recognition and reports errors via text-to-speech technology.

1.0 Introduction

The Java Audio Music-suite (J.A.M. for short) is a suite of applications that provides tools and audio processing utilities for musicians. Its core functionality includes a means whereby musicians who are geographically dispersed can play music together. An additional utility determines the actual musical notes from a piece of music using Fourier Transform techniques. All of these functions are complemented by a voice-enabled, animated help agent that takes in voice commands using speech recognition and reports errors via text-to-speech technology.

The basis of the application, the primary function, is a facility that allows geographically dispersed musicians to play live together over a network, or over the Internet using Java sockets with the Real Time Protocol (RTP).

The application has a number of functions, including the Fourier transform function. With this function, the user can input a melody into the system live, or as an audio file. This data is then passed through a Fourier algorithm, such that the data stream is converted into a group of frequencies. These frequencies are then be used to discern the musical notes that the frequencies represent. The notes found are then be used to get the key of the music and from that we get the accompaniment to be played with the music. This is both a valuable resource for songwriters who do not play instruments and a handy way for musicians to save time

while learning a new musical piece. At the moment we have implemented this function with slow mono-melodic audio streams (melodies with no harmonies). Differing degrees of audio compression and filters eliminate as much noise as possible so that the algorithm can be more effectively.

Another function is a help and error reporting system for the end user. We use Java speech technology for this. We complement this work with a macro-language defined in Backus-Naur Form (BNF) to underpin the voice activation system that allows a musician, whose hands are busy, to enter commands and make the choices presented by the help system. Error reports are returned through a speaking help agent.

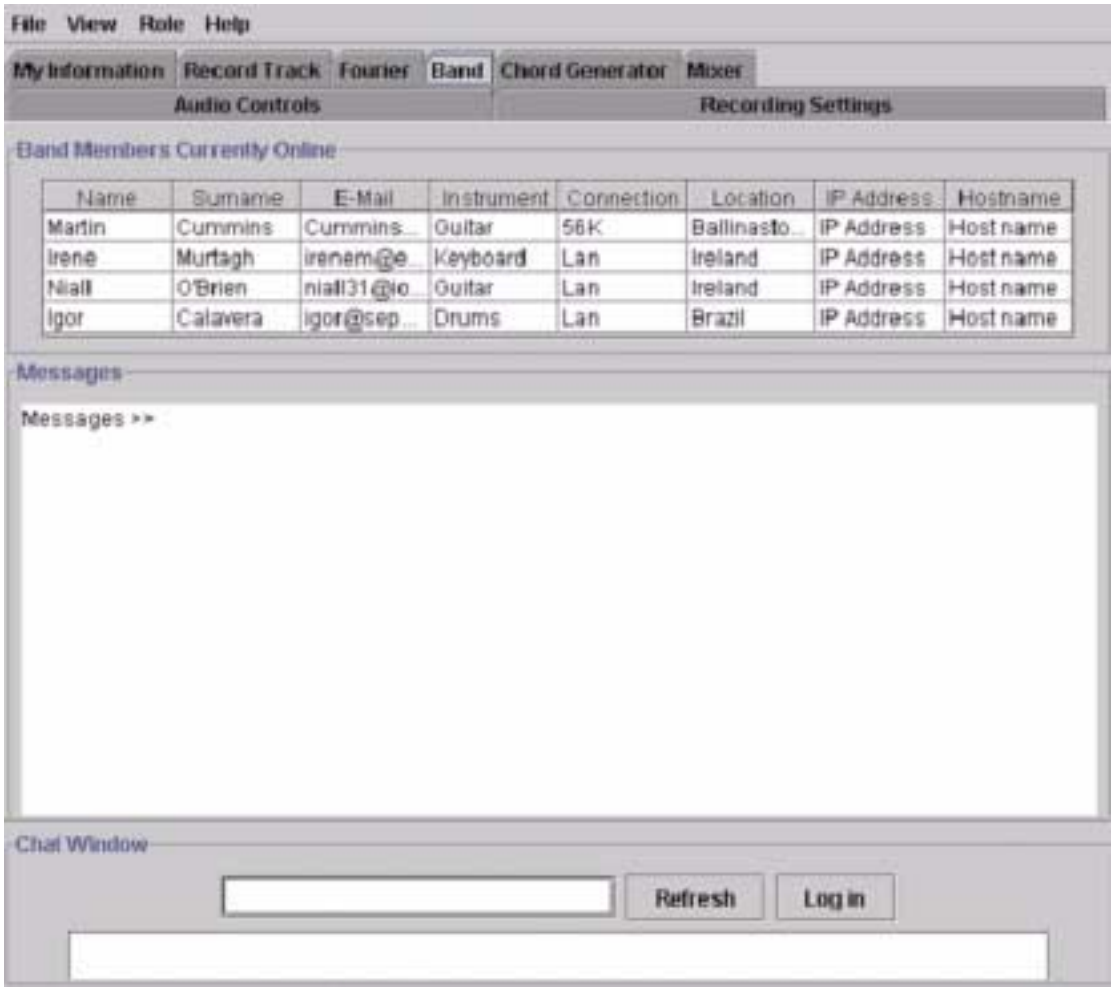


Figure 1: Screen shot of a user screen.

As a tool:

1. The software allows musicians who are geographically dispersed to play live together over a network or over the Internet. The audio data is synchronized so that the musicians are able to hear their own audio stream, in real time, but merged with the other musicians' audio stream.

2. Another user, who is not playing the music, is able to add sound effects and alter the respective volumes of the instruments remotely. This module, called the mixer module, has access to all of the audio streams and is able to manipulate the data for the users to hear. The mixer can record the data and manipulate it prior to the processing of the audio stream.
3. Each user has a basic version of the above on their own agent software that allows them to alter the sound and level of their own instrument.
4. We plan to include chord generators for guitars and keyboards (these will include a facility to hear the chords). The chords will be played from either MIDI data stored and reproduced through a sequencer, or from recorded audio.
5. A FFT (Fast Fourier Transforms) function discerns the notes from a melody and, from these notes, the key and chords to accompany that melody and the tempo of the music.
6. An intelligent help agent monitors the user's actions and provides help, suggestions and advice as appropriate. The user interacts with the agent by means of a voice-enabled speech recognition capability, and receives advice in the form of synthesized speech.

We describe some of the functions in more detail in the following sections.

2.0 The Virtual Band Function

2.1 Introduction

This function allows users who are geographically dispersed to play music live together over a LAN or through a suitable modem connection. This function is included as a core feature in the JAM suite. The primary technologies we utilise here are multi-threaded Java sockets, Java's Real Time protocol (RTP). Later we may use Java's Remote Method Invocation.

2.2 Difficulties for the Application

The primary difficulties are slow network speeds and difficulties synchronizing the users' input. This is further complicated by the time-critical nature of this kind of streaming multimedia application. We have considered a number of possible architectures and solutions to address these difficulties. We believe a client-server architecture to be the most effective because it helps to minimise the network traffic of the application. This minimisation is achieved by merging the audio streams received from the clients at a central point (the server)

so that the audio streams are merged before they are sent back as playback. These streams are captured and transmitted instantaneously using the Java Real Time Protocol (RTP).

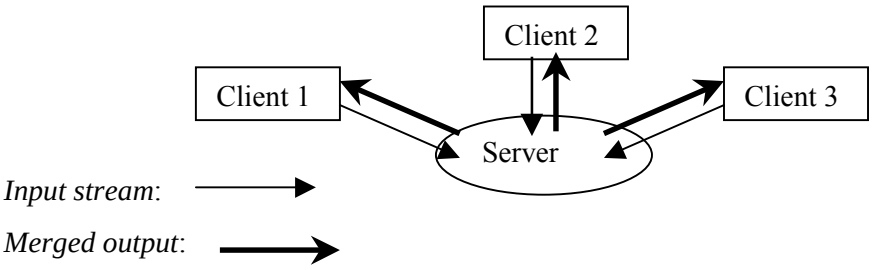


Figure 2: Overview of the Client-Server Architecture of the JAM suite

We have decided against the idea of using a *peer-to-peer* architecture because it would result in too much processing for each of the clients. For example, configured as a peer-to-peer topology with five users, each client would have to a), transmit its own stream and b), receive four others in real-time as well as c), capturing its own stream and d), playing all five.

2.3 The Buffer Idea

We have devised some techniques for handling slow connection speeds. One of these involves buffering the packets of audio data and numbering them on the transmitters side. We begin by sending only the first and fourth of the packets (for example). When their receipt is acknowledged we then send the second and third packet, but only if the server discerns that there is sufficient bandwidth to transmit the data. The server then has the option of deciding if the connection is fast enough at that time to warrant sending all four of the packets. If not, it will discard them and the process continues as previously. The client then has only the first and fourth packets. These are presented to the user with a "blank space" where packets two and three should be. This is obviously not ideal but it helps to ensure that there is minimal time lag in the playback of the audio streams on the receiver side. The data that was not sent in real-time could be sent to the user at a later stage and played as a backing track. The backing tracks are discussed shortly.

The basic steps underpinning the solution to the slow connection problem are:
Step 1: The audio stream is divided into a number of packets.
Step 2: Packets 1 and 4 are transmitted.
Step 3: An Acknowledgement is sent from the receiver on receipt of the "packets".
Step 4: The server discerns if it is viable to send the second and third "packets"

Step 5: Client leaves two blank spots (where packets 2 & 3 were) and plays one and four.

2.4 The "Overdub Idea"

Another idea we have devised is analogous to recording in a recording studio. It involves the users playing along to a backing track that is pre-recorded either in a previous session, or included as a default in the client. The users then agree between themselves on a backing track prior to the beginning of the recording session. They then all play along individually. The audio data is then sent to the server. These streams are amalgamated by the server and transmitted back to all the clients for their perusal. Clients then have the option of over-dubbing their parts afterwards. The server can keep copies of the original streams so that they may be included into later recordings of the same piece, or as backing tracks for later session.

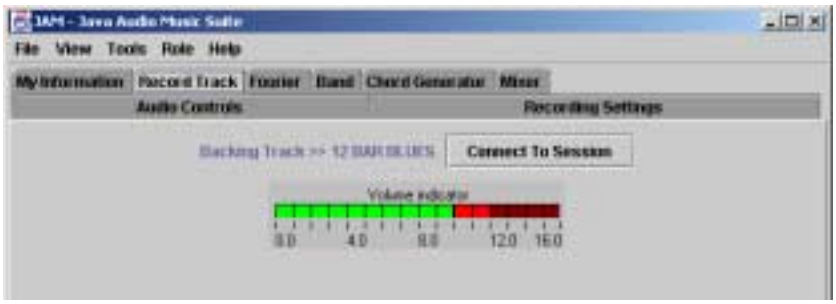


Figure 3: The figure shows the user recording a track accompanied by a backing track "12 bar Blues" .

3.0 The Fourier Transform

3.1 Introduction

In this section we describe the current applications for which Fourier Transforms are utilised. The Fourier transform decomposes or separates a waveform or function into sinusoids of different frequency which sum to the original waveform. It identifies or distinguishes the different frequency sinusoids and their respective amplitudes (Brigham: 4).

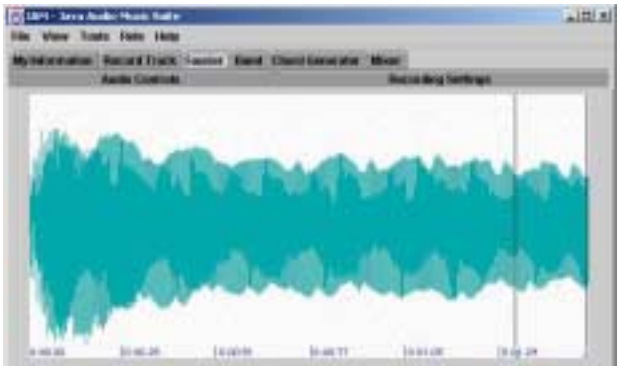


Figure 4: This figure shows the audio stream being inputted into the system prior to its transformation by the Fourier algorithm.

3.2 How We Use Fourier Transforms

We use Fourier transforms in the JAM suite to extract a precise frequency from an audio sample using spectral analysis. By comparing the discrete notes frequency and using a lookup table of note values and approximate frequencies, the application can then relay the musical notes to the user. This is done for all of the notes that are in the audio stream or sample.

3.3 Fourier Deployments

One of the deployments of the Fourier transforms is to provide the function performed by an electronic guitar tuner.

This involved:

- Creating a module to take in the audio samples in real time and place the audio data into a buffer.
- The sampled audio is then run through low pass IIR filters to clean it up before it is passed to the FFT transformation. The use of filters guarantee that only guitar range frequencies are involved in the spectrum and that the constituent noise in the sample is removed.
- The audio samples are processed with a Fast Fourier Transform to determine their spectral content.

Since the fundamental frequency of a guitar note contains the most energy, the FFT code is used to locate the frequency with the most energy from the buffer of samples. After all of the FFT data is examined, the frequency with the most power is be returned and this represents the current pitch of the guitar string.

3.4 Musical Pieces That Are Not Mono-Melodic

The next deployment using FFT builds on the guitar tuner application so that it functions over musical pieces that are not mono-melodic. This is achieved by processing an audio sample with a Fast Fourier Transform to determine its spectral content. However, this time we relay to the user the many musical notes in the audio sample. We achieve this by increasing the sampling rate and the number of transforms that will be required. This is

computationally expensive so at present we limit this function to operating over short pre-recorded musical pieces.

4.0 The Help Agent

4.1 Introduction

The purpose of the JAM Help agent is to provide users with context sensitive assistance at any stage during their utilisation of the program. It is also used for error management, providing the user with helpful information when an error has occurred, as opposed to the typical technical output of error information returned by the Java. User Interaction with the agent is by means of speech recognition and synthesis. The graphical interface for the help agent is designed in 3D Studio Max and Adobe Photoshop, and the plan is that the agent will react in accordance with the type of help that it is contextually required.

4.2 Contexts in Which the Help Agent will appear

The context in which help is provided will affect two areas:

4.2.1 Type Of Help

The JAM suite has many different stages of use with some steps being a necessary path in the achievement of other later steps. The help agent maintains a hierarchical representation of these steps and is context sensitive. If help is requested while the user is performing a step that leads to, for example, connecting to the mixer, then general mixer help will be made available to the user.

4.2.2 The Agents Appearance

We plan that the agent will have a set of pre-defined animations appropriate to the different types of help available. For instance, if the help were in relation to the Fourier Transform tool, then the agent would be holding a waveform in his hands as the tool is explained. We are still developing these sets of animations.

4.3 How We Implement The Speech Technology

The help agent interacts with the user by means of speech technology. This speech is provided through the use of IBM's implementation of the Java Speech API (JSAPI). The

speech is implemented through the use of the *runtimes* function provided by IBM's Via Voice. These *runtimes* contain the speech engines and data files that are integrated with this JAM suite application. The types of speech technology incorporated into the agent are speech recognition and speech synthesis.

4.3.1 Handling The Limitations Of Speech Recognition Technology

For the speech recognition part of the implementation we decided to limit the amount of commands available to the user to minimise he need for conversation level interpretation. The commands available to the user are basic discrete single words or phrases, examples of which include the simple "yes" and "no". We use a formal notation based on Backus-Naur Form to describe the syntax of a given set of commands. Due to the hierarchial nature of the commands and the context sensitivity provided by the agent we designed and built a small macro-language, and this is described in the same using the same BNF notation.

4.3.2 How We Will Implement Speech Synthesis

A decision had to be made as to whether we should use Java Speech Mark-up Language (JSML) for speech synthesis functionality included within our agent. JSML allows greater control and flexibility over Java speech by letting the programmer change the way in which words are said. We decided that the current version of the agent would not utilise this feature, but future versions might.

One of the attractions of Java is that utilising speech synthesis with the Java Speech API is relatively simple. One (merely) creates a synthesiser and invokes it's *speakPlainText*() method while handing it a String argument to be dictated.

4.4 The Graphical Side of the Help Agent

The character for the agent is initially based on the late Jimi Hendrix and is designed and rendered in 3D Studio Max through the use of Mesh Modelling. We design and export around 10 different AVI files, which are supported by the Java Media Framework (JMF v2.1.1). Through the use of JMF we create a player, which has a different AVI file running depending on the context of the help or during idle time. When the agent is speaking we also have a speech bubble onscreen that displays the text version of the help message being provided.

5.0 Conclusion

To conclude we feel that Java provides the ideal technology for this type of project because of the richness of the functionality within the language.

We feel that this application, as well as being an interesting and valuable learning experience for us, is a useful application of immediate interest to geographically dispersed musicians.

Java's cross platform compatibility is a very important for the success of the JAM suite, as not only will the users be from geographically dispersed areas but also they could be running the application on entirely different platforms. Java technology inherently allows for this eventuality.

Java's versatility and extensibility as a programming language proved very useful in the implementation of the JAM suite. Through the use of the Java Speech API and the Java Real Time Protocol, the benefits of developing with and deploying in Java become very apparent.

6.0 References

Lindsey, Craig A. (2000). *Digital Audio With Java*. Pearson Press UK.

// ***Interesting chat about Java sound and what it can do***
http://developer.java.sun.com/developer/community/chat/JavaLive/1999/jl1116.html
// ***Sound API programmers Guide***
http://java.sun.com/j2se/1.3/docs/guide/sound/prog_guide/title.fm.html
// ***Java Sound API Documents***
http://java.sun.com/products/java-media/sound/
// ***Interface Mixer taking audio lines and returning a single audio stream Clips***
// ***and real time***
http://www.chmsr.gatech.edu/java/api-js099/javax/sound/sampled/Mixer.html
// ***Java RTP (Real Time Protocol) page***
java.sun.com/products/java-media/jmf/2.1.1/solutions
// ***Fourier Transform Theory***
http://www.educatorscorner.com/experiments
// ***Brigham, Bracewell, Brault and White -Informative papers on Fourier Theory***
http://aurora.phys.utk.edu/~forrest/papers/fourier/
// ***Spectrum analysis using FFT***
http://www.dsptutor.freeuk.com/analyser/SpectrumAnalyser.html

GlobalCom

Declan Barber, Conor Gildea, Gavin Byrne
Institute of Technology Blanchardstown

Abstract

The objective of this developed application is to provide a company with a means of communicating with its employees no matter where they are physically located and what communication resources they may have at any particular time. The core focus of this application is to provide a Unified Messaging System using synchronous and asynchronous forms.

Introduction

Is there any application that currently provides this service?

There are many systems which currently offer services in a particular area of communications. None of these however offer an all-inclusive communications system. It is this niche in the market that we hope to aim our product at. In the market place to date, service providers provide some of the functionality that our application has to offer but the associated costs are high. What is novel about this project is it provides an autonomous enterprise platform with toll-free access for intranet and optimised extranet costs.

Why is this project a good idea and why is it useful?

This project is aimed at a niche in the market which will mean that for the near future the product will have no direct competition. With the increase in popularity of the internet, more and more people are seeking out more cost effective ways of communicating with each other. The internet provides a fast, efficient, and inexpensive medium of communicating. People do not always have access to a computer terminal connected to the internet and with this application it will provide them with a means of communication over the internet using their mobile phones. Mobile phone networks have become so advanced in the last couple of years that they provide a reliable world-wide medium of communication. Another aspect of this

application is that it provides a GPS telemetry service. This feature will allow the end-user to remotely track their GPS beacons and process the resulting telemetry data.

Typical Usage Scenarios:

1. *An employee out of reach of a computer terminal receives a critically important e-mail. The employee may not have an opportunity to read this for several days.*

This application will give the employee a means of accessing their e-mail without the need for a computer, but by using their mobile phone.

2. *An employee is away from the office and his resources are limited to a laptop and a mobile phone. The employee needs to be able to carry out every day functions as if he were actually connected to his company's local network.*

This application will provide the user with the means to communicate with their company's local network over a number of different mediums.

3. *As part of a distribution company's business, it is highly important that they know were their deliveries are at all times. This may include an employee who needs to know the location of a certain cargo at any time, whether they are physically present at a computer on the companies campus, or out in the field with nothing but a mobile phone.*

As part of this communications package, their will be the ability to obtain the location of any of the company's resources, i.e. delivery trucks, world-wide shipments, employee's cars, etc through the use of either a computer terminal connected to the internet or on a lesser level a mobile phone.

4. *A network administrator is looking for a new product that will cut down the need for expensive resources such as proprietary devices such as PBX's (Public Branch Exchange), cabling for voice only communications, and the need for employees trained solely in telephony skills that are needed to run these.*

This application will cut out the need for a separation between voice and data on a network. With this application the administrator has the ability to treat voice data in the same manner as regular data across the network, and eliminates the need for dedicated telephony hardware, cabling or personnel.

5. *A company wishes their employees to have the ability to hold a conference call over a corporate local area network or the internet.*

This application will provide users with the ability to hold one-to-one or conference calls with any other user of the system whether they are physically connected to their company's local network or whether they are connected to the internet anywhere in the world.

6. *A company wishes to have the ability to send information between their employees securely no matter where they are physically located.*

This application will give users the ability to encrypt their communications securely and efficiently.

7. *A computer wishes to have a web-based message board where any of their employees world-wide can add messages and read messages.*

This application will provide all users access to a central message board, to which anyone can add messages and read messages.

8. *A company needs the ability to provide real-time text-based chat between their employees using their corporate local network or the internet.*

This application will provide all users access to a virtual chat room to which anyone can hold a text-based conversation.

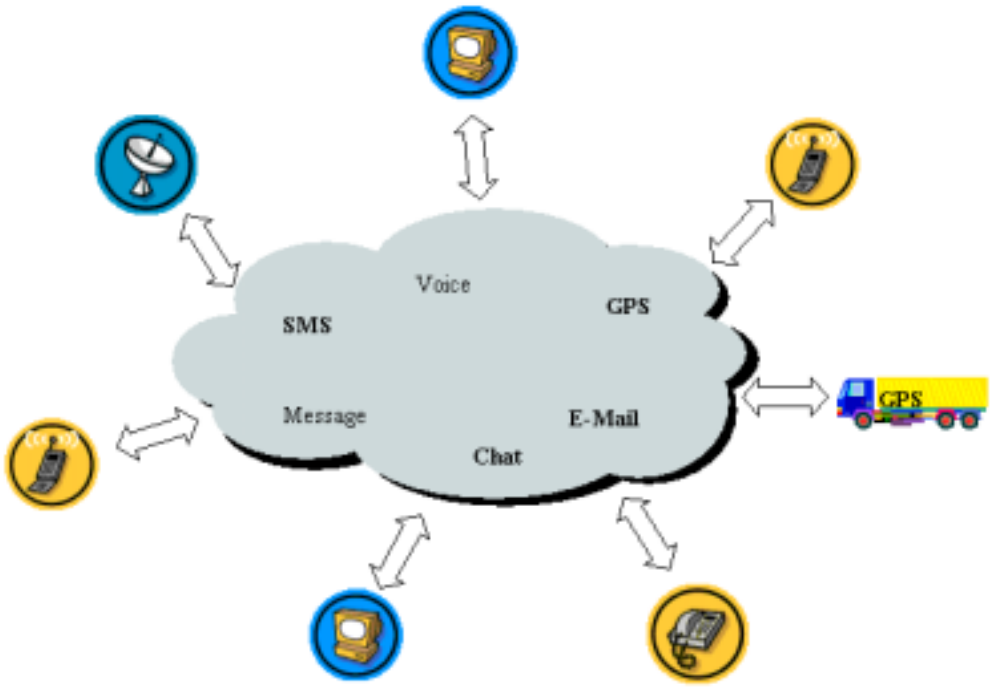
9. *A company wishes to have the ability to provide their employees with a secure company voice-mail system.*

As part of the application, there will be the option to send and receive secure voice-mails.

10. *A company wants the ability to convert easily from one medium to another providing them with a complete means of communications.*

The application will support the translation of communication mediums be it text or voice without the loss of information.

Application Topology



This application will provide communication through the following services:

- Text-based: These systems will incorporate functions for communicating through some sort of text-based application.
- Mobile: These systems will provide the ability to communicate with the main application through the use of a mobile phone.
- Internet: These systems will be accessible through any web browser connected to the internet.
- Intranet: All of the systems services will be accessible through a company's local network.
- Computer Telephony: These systems will give users the ability to use all the services usually related with normal telephones, from their computer terminals.
- The Global Positioning System: Theses systems will provide geographical Information about specific company resources.

Which aspects of JAVA technologies did we implement?

For our application we took full advantage of the various packages that JAVA has to offer. It is only through these packages, and some we developed ourselves, that it was possible to develop our system to its full potential. These include:

- Java Servlets
- Java Server Pages(JSP)
- Java Communications API
- Java Remote Method Invocation(RMI)
- Java Media Framework(JMF)
- Java Telephony API

Packages we developed during the course of our project:

- JAVA PGP encryption facility
- SMS package
- GPS package

How we used these technologies?

Java Servlets

The main benefit of using Java Servlets is that they are server-centric meaning that any changes that are made are propagated to the connected clients. These clients that are

connected can be stripped down terminals we low processing power accessing the system using a web browser. All of the processing is carried out on the server end and the processed requested is sent back to the requesting clients terminal. Java Servlets were very useful in generating dynamic web pages and retrieving information from a database. Servlets were also used in the PGP encryption system, were a client-run applet could make a call to Servlets running on the application server to pass encrypted data back and forth. For the SMS system, the use of Servlets enabled us to provide all users logged onto the system, the ability to send and receive SMS messages through a mobile phone module which was connected to the server.

JSP's

This was used for the parts of the system where JAVA was only needed to generate dynamic web pages as oppose to Servlets which were used for the systems requiring heavy processing. The main difference between Servlets and JSP's is that with JSP's JAVA code is embedded in the HTML content were Servlet classes are called from HTML pages, do some kind of processing and then generate web pages based on this processing. The main part of our application which used JSP's is the multi-roomed Chat-room.

JAVA Communications API

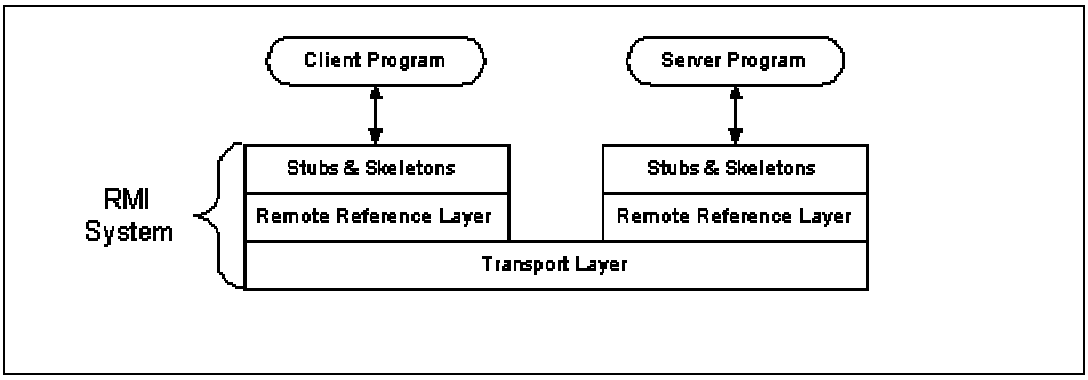
There are three levels of classes in the Java communications API: High-level classes like CommPortIdentifier and CommPort manage access and ownership of communication ports. Low-level classes like SerialPort and ParallelPort provide an interface to physical communications ports. The current release of the Java communications API enables access to serial (RS-232) and parallel (IEEE 1284) ports. Driver-level classes provide an interface between the low-level classes and the underlying operating system. Driver-level classes are part of the implementation but not the Java communications API. They should not be used by application programmers. This package provides an Enumeration of the available ports on the system. The static method `CommPortIdentifier.getPortIdentifiers` returns an enumeration object that contains a `CommPortIdentifier` object for each available port. This `CommPortIdentifier` object is the central mechanism for controlling access to a communications port, to resolve port ownership contention between multiple Java applications. Events are propagated to notify interested applications of ownership contention and allow the port's owner to relinquish ownership.

As an extension to this package we developed our own package that would handle all of the Serial IO communication with added respected to concurrent access to the Serial port. This consideration was very important, as the hardware would be connected to the server with the connecting clients competing for access to this device. The Serial IO can also be used when a telecommuter wishes to access the system remotely and there is no other form of communication other than the wireless mobile network. For the Serial IO package we decided to implement the singleton design pattern which prevents the class from being instantiated but all the method to be accessed directly and these methods are defined as been synchronised which means that once executed they have to finish and through the use of static variable it maintains that there is only ever one copy in memory. [1]

Java Remote Method Invocation (RMI)

RMI allows the code that defines the behaviour and the code that implements the behaviour to remain separate and to run on separate JVMs. This fits nicely with the needs of a distributed system where clients are concerned about the definition of a service and servers are focused on providing the service. Specifically, in RMI, the definition of a remote service is coded using a Java interface. The implementation of the remote service is coded in a class. Therefore, the key to understanding RMI is to remember that interfaces define behaviour and classes define implementation. With an understanding of the high-level RMI architecture, take a look under the covers to see its implementation. The RMI implementation is essentially built from three abstraction layers. The first is the Stub and Skeleton layer, which lies just beneath the view of the developer. This layer intercepts method calls made by the client to the interface reference variable and redirects these calls to a remote RMI service. The next layer is the Remote Reference Layer. This layer understands how to interpret and manage references made from clients to the remote service objects. The connection is a one-to-one (unicast) link. In the Java 2 SDK, this layer was enhanced to support the activation of dormant remote service objects via Remote Object Activation. The transport layer is based on TCP/IP connections between machines in a network. It provides basic connectivity, as well as some firewall penetration strategies. [2]

RMI Breakdown



Java Media Framework (JMF)

The Java Media Framework API (JMF) enables audio, video and other time-based media to be added to Java applications and applets. This optional package, which can capture, playback, stream and transcode multiple media formats, extends the multimedia capabilities on the JAVA platform, and gives the developers a powerful toolkit to develop scalable, cross-platform technology.

Java Telephony API (JTAPI)

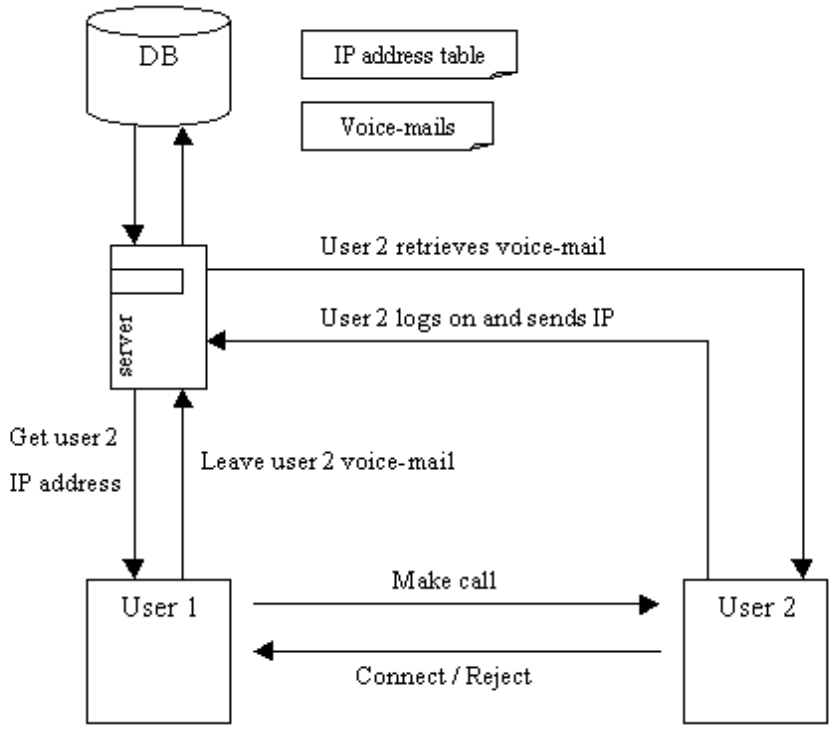
JTAPI is the set of classes, interfaces, and principles of operation that constitute a Java extension package in the *javax..** name space. JTAPI implementations are the interface between Java computer telephony applications and telephony services, whether those services are implemented as software, as in the case of a soft PBX, or hardware. JTAPI defines the access to one or more of the following areas of functionality: Call Control, Telephone Physical Device Control, Media Services for Telephony, and Administrative Services for Telephony.

The idea at the start of the project was to implement the telephony system as an applet which could be loaded up by the user from the web-page once they were logged in. However this proved more difficult than anticipated. Although the JTAPI model is described as being implemental as an applet, we found that any telephony program we developed needed access to certain system properties which are inaccessible to applets because of the use of the JAVA "sandbox" security model which runs on all JAVA Virtual Machines to stop programs downloaded from unknown sources causing malicious damage to the client's machine. We tried various techniques in trying to achieve this which included altering the client system's "java.policy" file to allow JAR files from our server's location all permissions to the client

machine (java.security.AllPermission). This was however only successful to the degree that we could run a JTAPI applet from appletviewer, which wasn't much good for the scope of the project. This is because appletviewer is not subject to the same security restrictions as applets loaded from a web browser. When this was unsuccessful, we tried to use signed JAR files to see if these would give our applet access to the required system properties. This however, was also unsuccessful.

Finally, we decided to implement a JTAPI application which would be downloadable from the server. This would run, but would only be fully functional once the user logged onto the main system. To do this we used JAVA RMI to pass encrypted messages between the client application and the main server where their information is stored. The IP address is retrieved by the server and used to access the client RMI application. This means that no matter were the user is logging in from, they can still access their missed/received/dialled calls history and any voice-mail they may have. Again the main idea behind this system is to have all components accessible from a web based e-mail style system incorporating folders for all systems-modules designed in a similar fashion. We also incorporated a secure telephone directory again using PGP encryption. All sending and receiving of audio data was controlled using the JMF. This toolkit provides multimedia features as described previously.

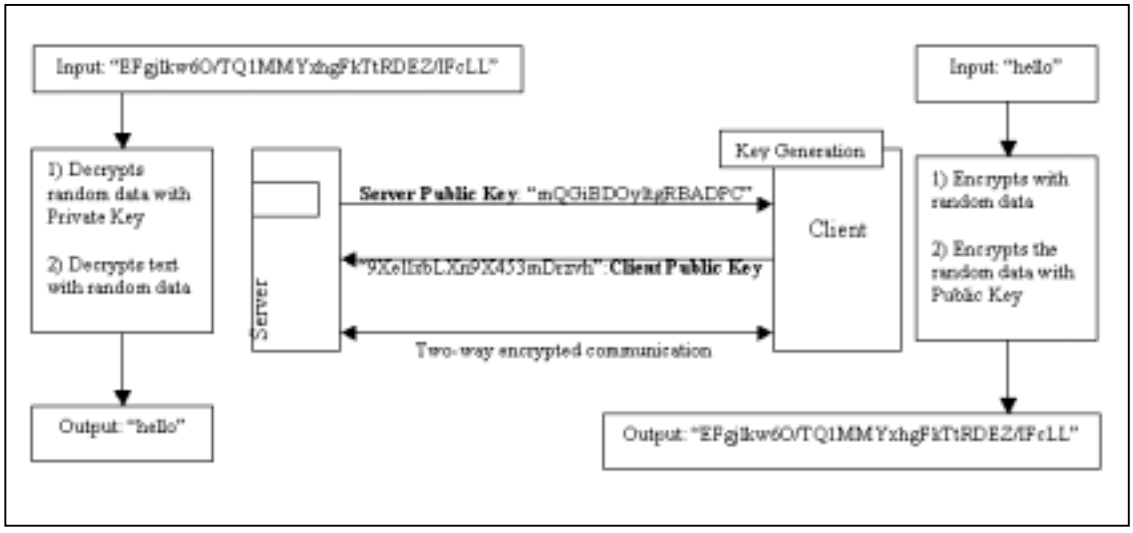
The basic Telephony Architecture:



JAVA PGP encryption facility

We developed a package for the encrypting and decryption of text using Public/Private key pairs. We developed a mechanism for generating the random data needed for key generation, which took in an array of mouse coordinates from an applet window. This overcame the problem of computer generated random numbers, which are never really random. This means that a user can generate key pairs on a session basis, thus doing away with the need to remember any passphrase which is usually used to encrypt the private key for storage. The only key paires stored are on the server, where firewalls prevent access to the private key. Although the passphrase for this key would be needed for it to be of any good to anyone, nobody should get the chance to try and guess the passphrase. The only keys that go out on the wire are the public keys of the server and clients which are of no use to anyone as the private key cannot be obtained from this. This improves the security element of the system, as the only totally secure encryption system is one that you have wrote yourself. With PGP encryption, the longer the text which is to be encrypted is, the more efficient the encryption becomes. Random data is used to encrypt the text, and then the public key is used to encrypt this random data which generates what is called a "Key Block". This is sent along with the encrypted text where, at the other end, the private key is used to decrypt this key block giving back the random data which can then be used to decrypt the text. [1]

PGP Architecture:



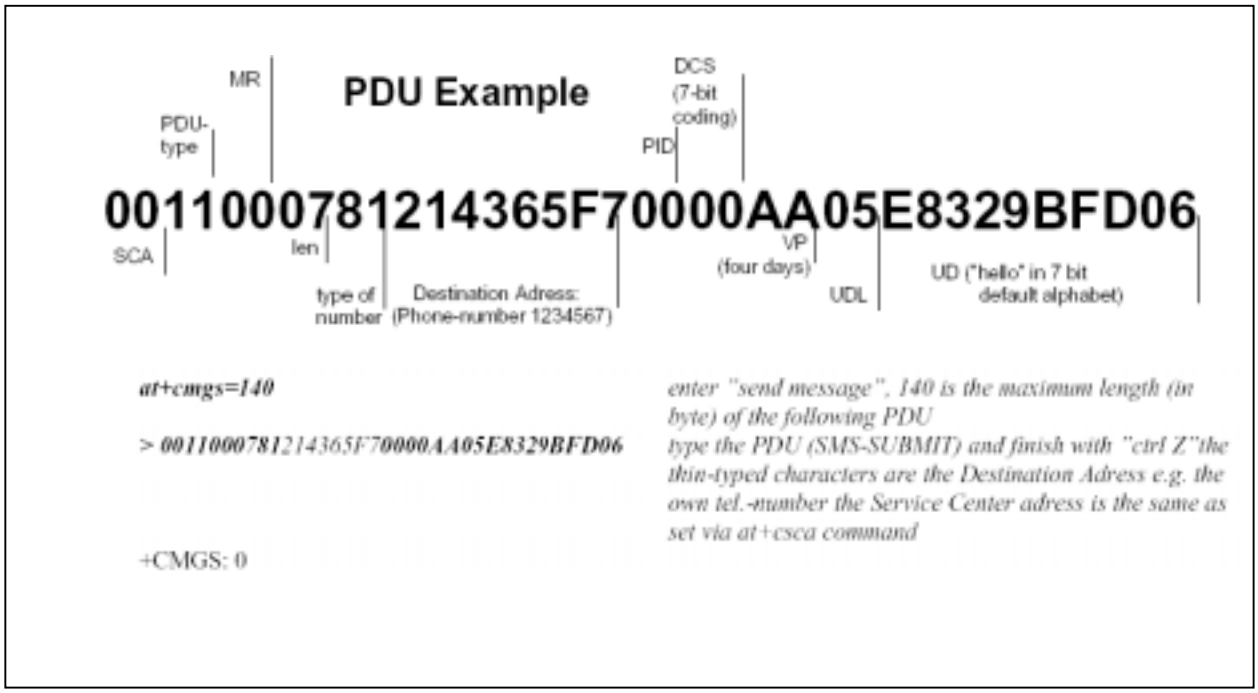
SMS package

Short messaging is a low-cost transport used to communicate with mobile stations (e.g. cellular phones, pagers) across wireless networks such as GSM networks. The Short Message Service, or SMS, is a bi-directional service for short binary and alphanumeric messages (up

to 160 chars). Messages are transported in a store-and-forward fashion via a Short Message Service Centre (SMSC). Mobile stations (MSs) can send and received messages to and from other mobile stations. In a short messaging context, the mobile stations are known as SMEs or Short Message Entities.

This package that we developed provides a toolkit of methods that can be used to send and receive SMS messages. This package will extend from the Java Comm API which is used to send and receive the SMS Protocol Data Unit (PDU). This functionality is already previously mentioned.

The SMS package will build on the existing functionality of SMS and will make a number of alterations. One of the limitations of SMS is that you can only send up to 160 characters (7-bit). The SMS package improves this limitation through the use of a compression algorithm that will allow the user to send up to 380 characters. This is only available from (PC-PC) communication or for future JAVA enabled phones. The compression algorithm bases itself on a Huffman style dictionary compression method. Other functionality that the SMS package provides is the translation from "text" to "quicktext". For example "I will see you later" translates to "I will c u l8r". There is also the ability to send flash messages that are only suitable for certain phone models. Another aspect of the system is that it will remove the messages from your "SIM" and store them in a database for future viewing. The system will provide a mechanism of translating from one message media to another from example message board thread to SMS message. [3]



GPS package

We have not got the GPS system working yet as we are still waiting to get the necessary hardware, i.e. GPS unit incorporating a GSM module. Once we have this, our aim is to provide users with access to positional tracking information about any vehicle(s) they wish to put one of these modules into. This will be done through data sent and received using data calls or SMS messages from these GPS modules to the GSM module connected to the server.

Overall Application Architecture:

<table>
<tr><td colspan="4">Application Layer</td></tr>
<tr><td>Java Api's</td><td>Tomcat WebServer</td><td>JTAPI</td><td>ODBC</td></tr>
<tr><td colspan="4">Java Virtual Machine (JVM)</td></tr>
<tr><td colspan="2">Platform (OS)</td><td colspan="2">Telephony Platform (Peer)</td></tr>
</table>

Conclusion

It was only through the use of JAVA as a technology was it possible for our application to reach its full potential. The computer world currently has many platforms, among them Microsoft Windows, Macintosh, OS/2, UNIX and NetWare; software must be compiled separately to run on each platform. The binary file for an application that runs on one platform cannot run on another platform, because the binary file is platform-specific. The Java Platform is a new software platform for delivering and running highly interactive, dynamic, and secure applets and applications on networked computer systems. But what sets the Java Platform apart is that it sits on top of these other platforms, and executes *bytecodes*, which are not specific to any physical machine, but are machine instructions for a *virtual machine*. A program written in the Java Language compiles to a bytecode file that can run wherever the Java Platform is present, on *any* underlying operating system. In other words, the same exact file can run on any operating system that is running the Java Platform. This portability is possible because at the core of the Java Platform is the Java Virtual Machine. It is this JVM that allows your application to run over heterogeneous system and allows for a "*Write Once, Run Anywhere*" capability.

Links & References

[1] http://java.sun.com
[2] http://www.oreilly.com
[3] http://www.dreamfabric.com/sms

Learning Through Dialogue (LTD) - A toolkit to support the process of planning for effective use of dialogue in learning

Matt Smith[1], John Cook[2] & Martin Oliver[3]

[1] *School of Informatics and Engineering,*
Institute of Technology at Blanchardstown, Dublin 15
[2] *Learning and Technology Research Institute,*
London Metropolitan University, UK
[3] ***Higher Education Research and Development Unit,***
University College London, London UK
Contact email: matt.smith@itb.ie

Abstract

This paper presents an implementation of a decision support system to help tutors think about ways of using dialogue to support learning. The approach adopted has been to develop a software toolkit around a knowledgebase of dialogue methods, to assist tutors in the reflection required during the planning and design of dialogue to support learning.

Keywords

Pedagogical issues, teaching/learning strategies, improving classroom teaching.

1. Introduction

The effective use of dialogue to support learning in Higher Education is complex and difficult. For educational practitioners with limited experience of dialogue methods, the task of designing (or redesigning) such teaching can be problematic and time consuming. The research described in this paper describes one strategy to support tutors in pedagogical use of dialogues. This project also works to contribute to wider issues about the development process for decision support toolkits, and the improvement of understanding about relative benefits of using dialogue to support different kinds of learning in different domains.

First the paper explores issues about the use of dialogue to support learning, followed by issues about software toolkits in general and their development. The paper then presents the design and development stages of "LTD – the Learning Through Dialogue Toolkit" itself, from initial design, through evaluation of a paper-based prototype, to full software implementation.

2. The use of dialogue to support learning

There are many techniques for using dialogue to support learning. New methods and modern interpretations of older ones have been encouraged by increased availability of Internet technologies and systems to support asynchronous communication. Examples of dialogue methods to support learning include Socratic approaches to repeated questioning, such as implemented the WHY computer-based system (Stevens, Collins & Goldin, 1982), and modern interpretations of Socratic dialogue, such as the theory of inquiry teaching proposed by Collins & Stevens (1991). Dialogue can be structured through formal rules for turn taking, such as the dialogue games described by Levin & Moore (1977). Others have explored apprenticeship models of the community of inquiry approach (the TAPS project, Derry 1992) and learning assistants to support collaborative dialogues (the MetaMuse system, Cook 2001).

There is certainly no single "correct" approach that will meet all pedagogic aims, and for educational practitioners with limited experience of dialogue methods, the task of designing (or redesigning) such teaching can be problematic and time consuming. Designing a computer-based system to support the planning for effective use of dialogue in learning is a non-trivial task, which we have tackled through the development of LTD, a software toolkit.

3. Toolkits to support decision making

A 'toolkit' is a software system to support a design process (in our case, the design of teaching that effectively incorporates dialogue methods). Toolkits support decision making based around an expert model of the design process – they provide a structure for decision making and can make recommendations, based on 'goodness of fit' between descriptions of problems elicited from the user and a knowledgebase of possible solutions. Oliver & Conole (1999) describe toolkits in more detail.

For the toolkit we describe in this paper we have followed the methodology for toolkit design (Conole & Oliver, in press), which can be summarised as follows:

(A) Identification of a suitable theoretical framework for design

(B) Toolkit specification (how can the range of options available at each stage be translated into a practical but flexible form of guidance for non-experts?)

(C) Toolkit refinement: evaluating prototypes (how useful and flexible is the toolkit?)

(D) Inclusion of user-defined features

(E) The development of shared resources

For the LTD toolkit the first two stages (A and B), and the first iteration of stage C have been described in detail previously (see Cook & Oliver, in press), and are only briefly summarised here. The focus of this paper is in presenting details a **second iteration of the prototyping step (C)** – the development of a full software prototype of the LTD toolkit.

4. (A) Identification of a suitable theoretical framework for design

No existing model could be found that described the process of comparing and contrasting different discursive formats for education. Therefore relevant case studies were considered and the following 5-stage model was derived:

- Identification of learning need
- Elicitation of learning objectives
- Elicitation of detailed description of task and context
- The filtering of options and recommendation of suitable approaches
- Selection, investigation and adoption of a suitable approach by the user

5. (B) Toolkit specification: how can the range of options available at each stage be translated into a practical but flexible form of guidance for non-experts?

A prototype toolkit was designed around the five decision-making steps identified in the theoretical framework for the design task (steps 1 to 5 above). Descriptions are provided to the user at each decision-making step, and the user is required to perform an activity/make a choice, while able to interrogate options and knowledgebase contents. The toolkit approach requires the organisation of information in 'layers', allowing users to engage deeply with content/decisions they find important, and able to quickly skip over, or bypass completely, steps with less relevance to their particular teaching requirements.

A knowledgebase was designed to contain details of different kinds of dialogue methods (such as court of law, structured debates, and Socratic dialogue). To assist users in finding a good 'fit' between their teaching aims and available dialogue methods a 'knowledge space' was developed to differentiate between features of different dialogue methods. This knowledge space comprised the following set of criteria identified as useful for distinguishing between dialogue methods:

- Whether or not it is important to reach a 'right' answer.
- Whether the method emphasises collaboration or competition.
- The duration required for a dialogue of this type.
- The numbers of participants required.
- Issues of power relationships within the discussion.

These descriptors were adopted on their pragmatic value in discriminating between alternatives in a way that is likely to identify useful approaches. Since these descriptors are not necessarily independent of each other, nor exhaustive etc., part of the evaluation of our toolkit will require validation of this knowledge space.

The structure of the dialogue methods knowledgebase was based on this set of descriptors. Part of the design of the interactive aspects of the prototype toolkit involved asking the user to rate each of their learning objectives using this same mapping used to represent methods in the knowledgebase. For the toolkit design the questions asked and ranges of answers permitted were as follows:

- Is there a right/wrong answer? (Options: there is an absolute answer, there are criteria for 'right' answers, very open/free.)
- Should the dialogue promote competition or collaboration? (Options: there will be clearly identified 'winners' and 'losers', the dialogue will be highly competitive, some people will do better than others, success will only be judged in terms of the group as a whole, credit will be given for listening to others and drawing on evidence.)
- What will the duration be? (Options: hours, days, months.)
- What will the group size and level of tutor support be? (One interval, e.g. "25-30 students", one integer.)
- Issues of power (who defines rules and roles?) (Options: defined and assigned by the tutor, negotiated between tutor and participants, determined by participants, no formalisation of power/roles.)

6. (C) Toolkit refinement: evaluating prototypes

A first, 'paper' prototype was developed and evaluated with users through the "Wizard of Oz" approach (see Cook & Oliver, in press). This paper-based prototype toolkit was tested with teachers/tutors, and evaluated to assess its suitability, ease of use, flexibility and relevance. The result was that the following decisions were made to refine the toolkit design, based upon the results of the prototype evaluation:

- reduce number of steps by requiring the user to focus on a single learning objective at a time (thus there are now 4 steps, with the new step 2 combining the original steps '2 Elicitation of learning objectives' and '3 Elicitation of detailed description of task and context')
- provide a range of detailed examples for each user input, so the user understands appropriate type of entry

Other design decisions implemented in the refined system include:

- allow non-linear navigation – so user can visit decision steps in any order, trying out different choices ("what if" scenario experimentation) and seeing how recommendations change
- knowledgebase entries rank criteria on a Real number scale from +1.0 to –1.0, the system returns 'goodness of fit' to the user in terms of an average match between user choices and knowledgebase entries in the range +100.0% to –100.0% (so a simple measure of closeness of match of alternatives is available, rather than simply the rank position of alternatives)

7. Second iteration of step (C) Toolkit refinement: evaluating prototypes

Building upon the findings of the evaluation of the paper-based prototype, a full software prototype has been developed and is the focus of this paper. The following diagram (Figure 1) presents a simple overview of the toolkit and interactions:

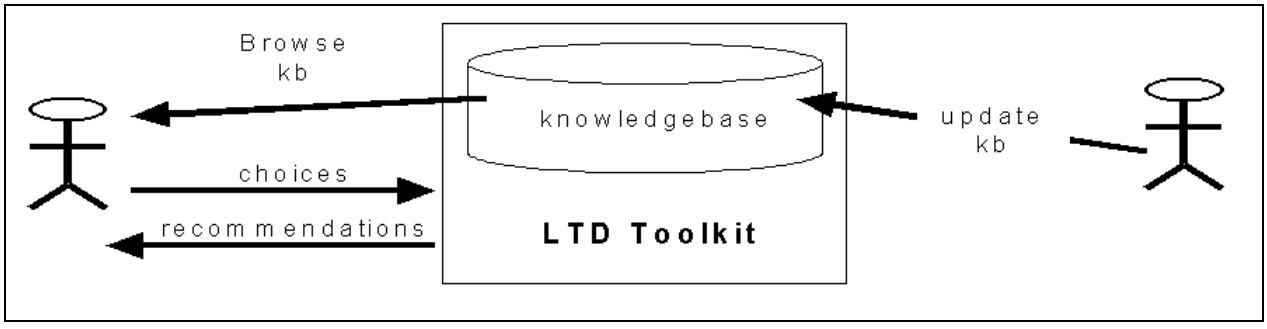


Figure 1: Overview of LTD toolkit.

Figure 2 illustrates a screenshot of the toolkit home page:

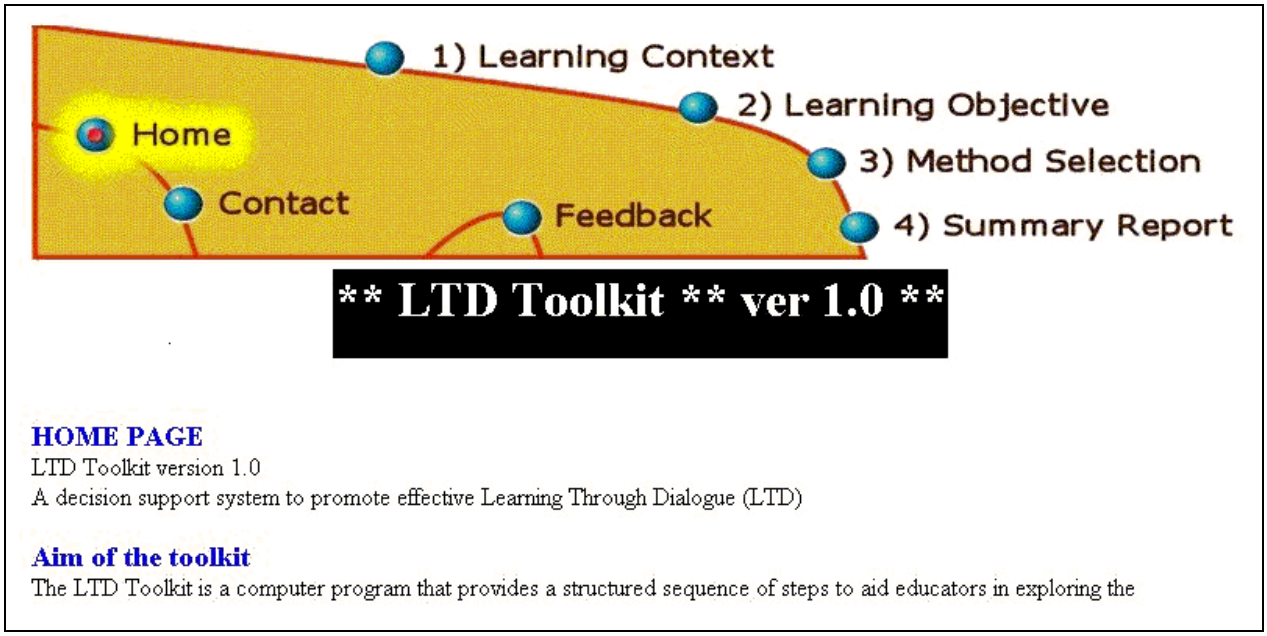


Figure 2: The LTD home page

Navigation is performed by clicking on the image map at the top of the window, or on the buttons on either side of the image map.

7.1 Step 1 – Learning context

Step 1 involves the entry of details about the learning context. The screenshot below (Figure 3) illustrates entry of details for Step 1.

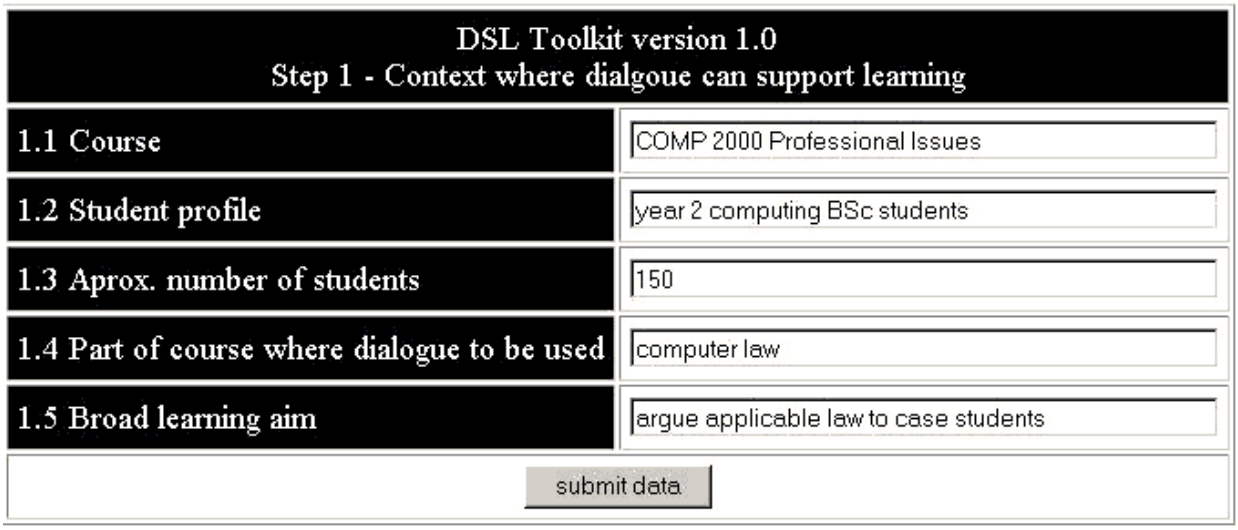


Figure 3: Learning context (Step 1).

7.2 Step 2 – Learning objective rating

Step 2 supports the user in describing in detail features of the learning outcome that the teaching/learning is to support. Textual descriptions of the learning objective, and how its achievement is to be judged are provided by the user. The user then rates their learning objective in terms of the knowledge map that encodes the knowledgebase entries. Examples of two of the questions, and structured choices for each have been implemented are as follows:

B: Degree of encouragement of collaboration

- "(doesn't matter)",
- "Individual competition required",
- "Individual competition encouraged",
- "Distinct contributions to group encouraged",
- "Collaboration within group encouraged"

E: Definition of power roles

- "(doesn't matter)",
- "Tutor assigns roles & rules",
- "Tutor and Participants negotiate roles & rules",
- "Participants assign roles & rules",
- "There is no formalisation of roles & rules"

The figure below is a screen shot of example entries and choices for Step 2.

DSL Toolkit version 1.0 Step 2 - Details of learning objective	
2.1 Learning Objective (LO)	identify / apply compute misuse law
2.2 How to judge achievement of LO	written report and oral argument
2.3A Openess of domain	Open domain
2.3B Degree of encouragement of collaboration	Collaboration within group encouraged
2.3C Duration of learning/assessment experience	Over several months
2.3D1 Group size	Exactly 6
2.3D2 Level of tutor support	Exactly 1 tutor
2.3D3 Scope of tutor support	The tutor(s) given in D2 above have to spread their tutorial support over all groups
2.3E Definition of power/roles	Tutor assigns roles & rules
submit data	

Figure 4: Details of Learning Outcome (Step 2).

7.3 Step 3 – Method selection

Step 3 is where the toolkit ranks each method in the knowledgebase against the rating choices made by the user for the learning objective. Rankings can range from +100% to –100% - these figures are based on the mean of the closeness of the match between each criterion entry for a dialogue method and the actual ratings entered by the user. An example of the

presentation of ranked dialogue methods is illustrated in the following screenshot (at the time of writing only 'court of law' has a full knowledgebase entry, the knowledgebase is being populated with a range of dialogue methods over the next few months).

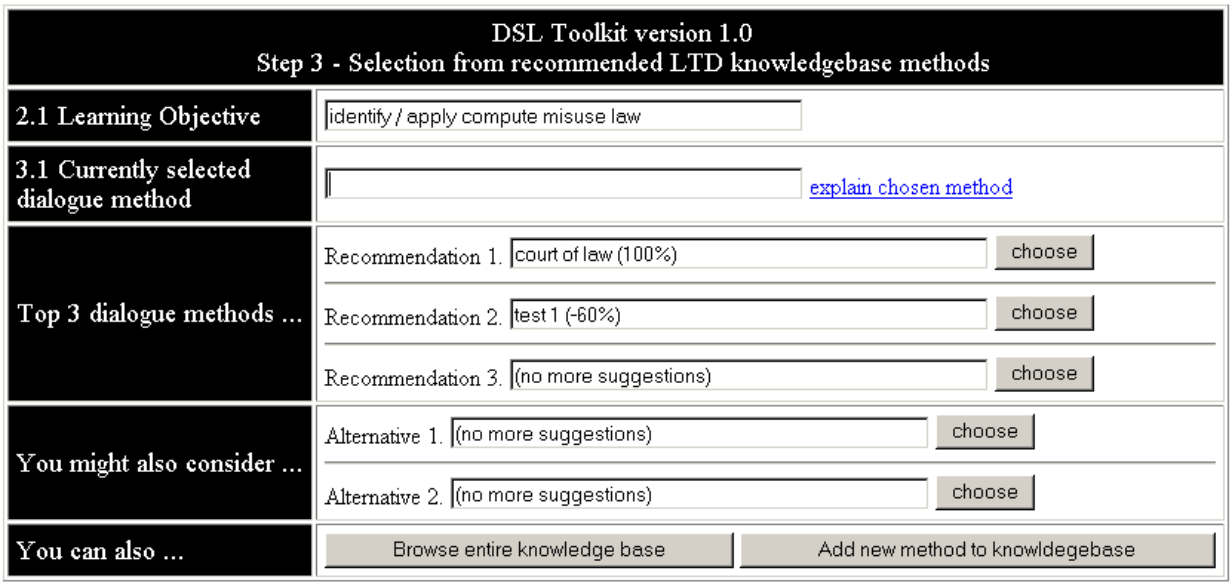


Figure 5: Method Selection (Step 3).

A simple explanation facility is provided, providing details of the percentage match of each criterion entered by the user with the selected knowledgebase method. This is illustrated in Figure 6 below.

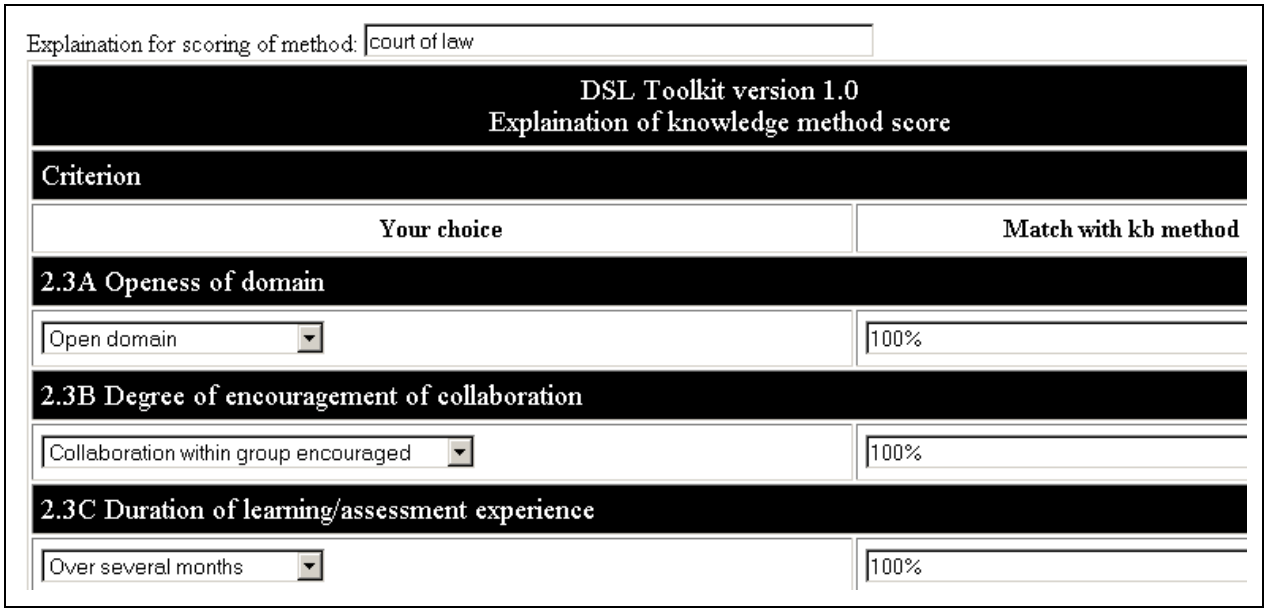


Figure 6: Excerpt from explanation of selected method score.

7.4 Step 4 – Summary Report

Step 4 collates all the entries by the user as a summary report for printing or saving to file. An example screenshot from the toolkit presenting a summary report is as follows:

DSL Toolkit version 1.0 Step 4 - Summary report	
1.1 Course	COMP 2000 Professional Issues
1.2 Student profile	year 2 computing BSc students
1.3 Aprox. number of students	150
1.4 Part of course where dialogue to be used	computer law
1.5 Broad learning aim	argue applicable law to case students
2.1 Learning Objective (LO)	identify / apply compute misuse law
2.2 How to judge achievement of LO	written report and oral argument
2.3A Openess of domain	Open domain
2.3B Degree of encouragement of collaboration	Collaboration within group encouraged
2.3C Duration of learning/assessment experience	Over several months
2.3D1 Group size	Exactly 6
2.3D2 Level of tutor support	Exactly 1 tutor
2.3D3 Scope of tutor support	The tutor(s) given in D2 above have to spread their tutorial support over all groups
2.3E Definition of power/roles	Tutor assigns roles & rules

Figure 7: Summary Report (Step 4).

9. Conclusions and further work

The notion of principled design was one of five current strategic issues identified by Harasim (2001). In this paper we have described a software toolkit that allows practitioners to use dialogue in learning in an educationally driven way. We feel that LTD provides a simplification of the complex process of effectively incorporating dialogue into learning. The tool is flexible in that it allows users to skip over sections they feel irrelevant and engage more deeply with content they find important. The non-linear navigation supports easy identification of alternatives and the browsing of their knowledgebase entries.

The contribution of this paper is a report on the implementation of the LTD software prototype. The LTD system will, we hope, provide a useful decision support tool for teachers/tutors in all sectors of learning – enabling them to plan their approach to incorporating dialogue in their own learning context.

Future work will test our claims with user evaluations of the software prototype. We will then progress to the final two steps (D and E) in the toolkit design methodology.

10. Location of the toolkit and invitation to contribute

The toolkit can be found at the following URL.

`www.itb.ie/staff/mattsmith/`

The toolkit is implemented in HTML and Javascript.

References

Collins, A. & Stevens, A. L. (1991). A Cognitive theory of Inquiry Teaching. In P. Goodyear (Ed.), *Teaching Knowledge and Intelligent Tutoring,* 203-230. Norwood, NJ: Ablex.

Conole, G. & Oliver, M. (in press). Embedding Theory into Practice Using Toolkits. *Journal of Interactive Media in Education*.

Cook, J. (2001). Bridging the Gap Between Empirical Data on Open-Ended Tutorial Interactions and Computational Models. *International Journal of Artificial Intelligence in Education*, 12, 85-99.

Cook, J. & Oliver, M. (in press). Designing a toolkit to support dialogue in learning. *Computers and Education.*

Derry, S. (1992). Metacognitive Models of Learning and Instructional Systems Design. In M. Jones & P. Winne, (Eds.), *Adaptive Learning Environments: foundations and frontier.* Hamburg: Springer-Verlag.

Harasim, L. (2001). The Future of Learning (Keynote Lecture). In *CAL 2001 Learning across the ages–looking back and looking forwards, Abstract Book,* p. 39. Elsevier Science.

Oliver, M. & Conole, G. (1999) *From Theory to Practice: A Model and Project Structure for Toolkit Development*. BP ELT Report No. 12, University of North London.

Stevens, A. L., Collins, A. & Goldin, S. E. (1982). Misconceptions in Students' Understanding. In D. H. Sleeman & J. S. Brown (Eds.), *Intelligent Tutoring Systems*, 13-24. London: Academic Press.

A Java Framework for Computer Vision

Stephen Sheridan
Institute of Technology Blanchardstown
Stephen.sheridan@itb.ie

Abstract

This paper outlines a framework implemented entirely in Java that attempts to give students exposure to computer vision systems from a practical standpoint. Various tools and technologies are introduced that will allow a student to acquire an input image through a WebCam, extract useful information from that input image and finally, attempt to make sense of the input.

1 Introduction

The field of computer vision is a diverse and interdisciplinary body of knowledge and techniques that attempt to understand the principles behind the processes that interpret signals from various sensors. Computer vision is often studied at an undergraduate level as part of the wider area of Artificial Intelligence or as a topic in its own right. Much of the work in teaching a subject like computer vision is theory based, as building complete vision systems takes an enormous amount of time and expertise. The problem that lies at the heart of computer vision is that of recovering information about the world from images. As with human vision, computer vision starts with the problem of image acquisition. Nature has provided humans with an excellent image acquisition device, the eye. Light enters the eye through a lens and is focused on a special layer of light sensitive cells at the back of the eye called the retina. From there, the light signals are converted to electrical impulses that travel down the optic nerve to an area of the brain called the striate cortex [1]. In comparison to what we know about the human eye, little is known about the function of the parts of the brain dedicated to vision. By using the human visual system as a model for computer vision we can break the problem down into three basic stages.

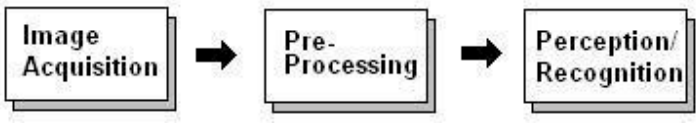


Figure 1: Simplification of stages for computer vision

The simplification of the computer vision problem shown in figure 1 does not suggest that this is what happens in humans, however it does provide a useful decomposition of work for a computational model.

2 Image Acquisition

Image acquisition is usually the first problem that students encounter when starting a computer vision project. Five or more years ago, this may have been more problematic because of the limited availability and high cost of devices such as digital video cameras and WebCams. Today however, the starting point for a computer vision project is much cheaper. Because of the growing interest in video conferencing, highly functional WebCams have become widely available and can be bought for very reasonable prices. From a practical standpoint, WebCams provide a quick and easy means of capturing images as they conform to the TWAIN specification. TWAIN defines a standard software protocol and API for communication between software applications and image acquisition devices [2]. The latest TWAIN specification can be downloaded from http://www.twain.org. All image acquisition devices that comply with the TWAIN specification can be accessed programmatically through the TWAIN API. The TWAIN drivers and API usually reside on the host operating system as native code, therefore, a Java wrapper API must be written to make the TWAIN API available to Java programmers. Luckily enough for Java programmers, a Java TWAIN API can be downloaded from http://www.gnome.sk and at the time of this writing the API is free for non-commercial use.

2.1 Using the Java TWAIN API

Once the Java TWAIN API has been downloaded and installed, students can begin to write simple programs that capture images from TWAIN compliant devices such as Scanners and WebCams. The TWAIN API is easy to use and does not require the student to have any prior knowledge of the actual hardware being used.

Listing 1 shows the main steps involved in capturing an image from a WebCam.

```
// Imports
import SK.gnome.twain.*;

// Step 1:Create a new TWAIN Object
Twain twain = new Twain();
// Step 2:Allow the user to select a data source (Scanner,WebCam,...)
twain.selectSource();
// Step 3:Pop up Twain user interface.
twain.setVisible(true);
// Step 4:Grab an image from the TWAIN device
image = Toolkit.getDefaultToolkit().createImage(twain);
```

Listing 1: Code sample showing steps to capture an image from a WebCam

In listing 1 the first step simply creates a Java TWAIN object that is used to control the image acquisition device. By calling the *set*XXX methods provided by the TWAIN class, the programmer can set various properties of the image acquisition device. A detailed listing of the available *setXXX* methods can be found in the Java TWAIN API documentation and further information about each of the device properties can be found in the TWIAN specification document [3]. Step 2 of listing 1 allows the user to select an image acquisition source. This is a useful feature as more then one TWAIN compatible source may be attached to the system. Figure 2 below, shows the source selection window that appears when this method is called.

Figure 2: TWAIN source selection window

Step 3 of listing 1 sets the visible property of the TWAIN object to *true,* causing a vendor specific UI to appear that allows the user to manipulate the image before it is captured. In some cases where this is not desirable, the value passed to this method can be set to *false* and the UI will not appear. However this feature will only work if the image acquisition device supports the TWAIN specification version 1.6 or higher. The code in step 4 of listing 1 shows how an AWT IMAGE object is returned from the TWAIN device. When this line of code is executed the TWAIN UI will appear and user can configure the image before it is captured. Figure 3 shows an example of the type of UI that is displayed when the *setVisible* method is used with a value of *true.*

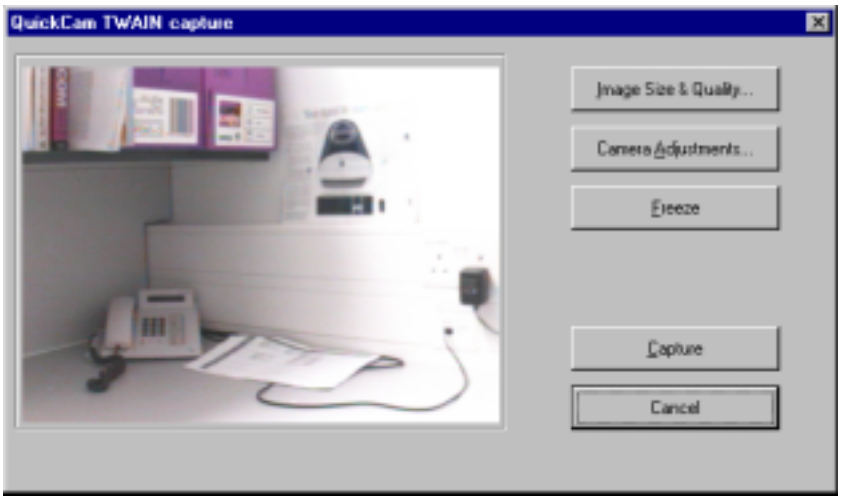


Figure 3: Example of TWAIN user interface

Once a valid AWT IMAGE object is returned from the TWAIN device, standard Java image handling techniques can be used to display or manipulate the image further. At this point in the computer vision process the pre-processing stage must take over and extract some useful information from the image.

3 Pre-processing

The pre-processing stage is concerned with extracting useful information from the image that can be passed on to the final perception or recognition stage. Many of the operations carried out at this stage are based on standard image processing techniques such as filtering and edge detection. These image-processing techniques have huge applications in the area of computer vision. By applying filters to an image we can change the colours used in an image, change the image contrast, brighten or darken the image or blur the image. Blurring is important because it can improve the results of the perception/recognition phase by reducing the amount of noise in an image. Images can contain noisy data for many different reasons, for example, there may be dirt on the camera lens, the lighting conditions may not be adequate or the quality of the image acquisition device may not be suitable for a computer vision application. Edge detection is really just a special type of filtering that can be applied to an image to find abrupt changes in pixel intensity values. This type of filtering is very important in the area of computer vision, as edges are what define the shape and structure of our world [4]. The implementation of the image processing techniques mentioned above could seen as a project in its own right and seeing as though there are many libraries that provide this functionality already, it is unnecessary to ask students to re-invent the wheel. This however, does not mean that having a deep understanding of the above image processing techniques is

unimportant; it simply means that students can participate in practical computer vision projects without having to get bogged down in implementation details. The JAI API (Java Advanced Imaging API) is one such library that can be used to carry out various image-processing operations and can be downloaded for free from http://java.sun.com/products/java-media/jai/.

3.1 Using the Java Advanced Imaging API

JAI is a set of classes that allow sophisticated high-performance image processing to be incorporated into Java applets and applications. JAI implements a set of core image processing capabilities including:

- Filtering
- Edge extraction
- Statistical operators
- Image I/O
- Region of interest control
- Logical image operators
- Fourier transforms
- Image interpolation
- Histogram operators
- Image tiling

The first stage in using the JAI API is to create a parameter block. Parameter blocks are used to hold information on how the image is to be processed. The code listing below shows how a parameter block can be constructed to blur an image.

```
// Imports
import javax.media.jai.*;
import java.awt.image.renderable.ParameterBlock;

// Step 1: Setup blur kernel
float[] blurData      = {0.0F,          1.0F/ 8.0F,   0.0F,
                         1.0F/ 8.0F,   4.0F/ 8.0F,  1.0F/ 8.0F,
                         0.0F,          1.0F/ 8.0F,   0.0F
};
// Step 2: Create a new kernel object using the blurring data
KernelJAI kernel = new KernelJAI(3,3,1,1,blurData);

// Step 3: Create a param block with the source image and blurring kernel
ParameterBlock paramBlock = new ParameterBlock();
paramBlock.addSource(pimage);
paramBlock.add(kernel);

// Step 4:Call the appropriate JAI create method using the paramBlock
PlanarImage processedImage = JAI.create("convolve",paramBlock);
```

Listing 2: Code sample showing how to construct a parameter block for blurring

Step 1 of listing 2 creates a mask that will be used to average the pixel values in the source image. Different values can be used here to create various effects or levels of blurring. Figure 4 shows the effects of applying two different levels of blurring to a source image that contains noise. Step 2 creates a KernelJAI object; the KernelJAI class takes the values of the blurData and creates a 3x3 matrix that will be used during the blurring process. Step 3 creates a parameter block object and adds the source image and the blurring kernel. Step 4 makes a call to the JAI.create method with the appropriate operation name. When this method is called, the image added to the parameter block is processed using the given values. The JAI.create method returns a new PlanarImage that is the result of processing the source image with the values contained in the paramBlock object.

Figure 4: (A) source image with noise, (B) image blurred once (C) image blurred twice

Using the JAI API to detect edges follows almost the same steps as the blurring example in listing 2. The only difference is that instead of constructing a kernel that will average the intensity values in the image we must construct horizontal and vertical kernels that will detect abrupt changes in pixel intensity. Many different types of edge detection masks can be used for this purpose [5]; some popular masks are shown in figure 5.

The process of constructing the edge detection kernels is exactly the same as before, except this time it is necessary to construct both horizontal and vertical masks that can be added to the parameter block. Listing 3 shows how this can be done for the Sobel masks shown in figure 5.

0.0	**-1.0**	**0.0**
-1.0	**4.0**	**-1.0**
0.0	**-1.0**	**0.0**

Simple horizontal

0.0	**0.0**	**-1.0**
0.0	**1.0**	**0.0**
0.0	**0.0**	**0.0**

Roberts horizontal

1.0	**0.0**	**-1.0**
2.0	**0.0**	**-2.0**
1.0	**0.0**	**-1.0**

Sobel horizontal

0.0	**-1.0**	**0.0**
-1.0	**4.0**	**-1.0**
0.0	**-1.0**	**0.0**

Simple vertical

-1.0	**0.0**	**0.0**
0.0	**1.0**	**0.0**
0.0	**0.0**	**0.0**

Roberts vertical

-1.0	**-2.0**	**-1.0**
0.0	**0.0**	**0.0**
1.0	**2.0**	**1.0**

Sobel vertical

Figure 5: Selection of popular edge detection masks

```
// Step 1: Setup horizontal & vertical masks
float[] sobel_h_data  = { 1.0F,  0.0F, -1.0F,
                          2.0F,  0.0F, -2.0F,
                          1.0F,  0.0F, -1.0F
};
float[] sobel_v_data  = { -1.0F,  -2.0F, -1.0F,
                           0.0F,   0.0F,  0.0F,
                           1.0F,   2.0F,  1.0F
};

// Step 2: Create horizontal & vertical JAI kernel objects
KernalJAI kern_h = new KernelJAI(3, 3, sobel_h_data);
KernalJAI kern_v = new KernelJAI(3, 3, sobel_v_data);

// Step 3: Create a param block with the source image and Sobel kernels
ParameterBlock paramBlock = new ParameterBlock();
paramBlock.addSource(pimage);
paramBlock.add(kern_h); paramBlock.add(kern_v);

PlanarImage processedImage = JAI.create("gradientmagnitude", paramBlock);
```

Listing 3: Code sample showing how to create Sobel masks

Once the JAI kernel objects and parameter block have been created, the JAI.create method can be called to process the source image. This time the JAI.create method is called with the operation name "gradientmagnitude", this will cause both the horizontal and vertical masks to be used during the operation. Figure 6 shows the results of using the Simple, Roberts and Sobel masks respectively.

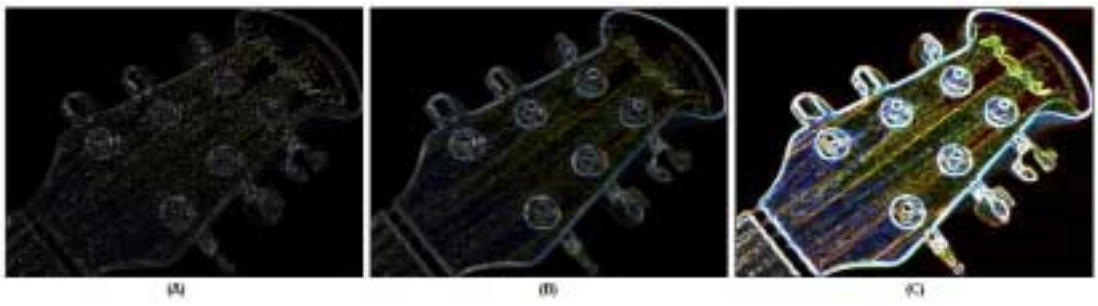


Figure 6: (A) Simple mask, (B) Roberts mask (C) Sobel mask

3.1 Conversion to raw data

The final phase of the pre-processing stage is to convert the edge image into raw data. This is necessary because the perception/recognition stage cannot deal directly with the images produced by the JAI API. There are many ways in which this can be done; one simple approach is to apply a threshold operator on the edge image. Since each pixel in the edge

image has an intensity value between 0 and 255, we can isolate those pixels that are part of an edge and those that are part of the background. For example, the edges shown in figure 6 diagram C) are primarily made up of pixels whose intensity values are greater then 150, so it is possible to construct a raw data file where a pixel that is higher then the threshold value of 150 is represented by a 1 and a pixel that is lower then the threshold value is represented by a 0. Figure 7 shows how this can be done for an edge image of a handwritten alphabetic character.

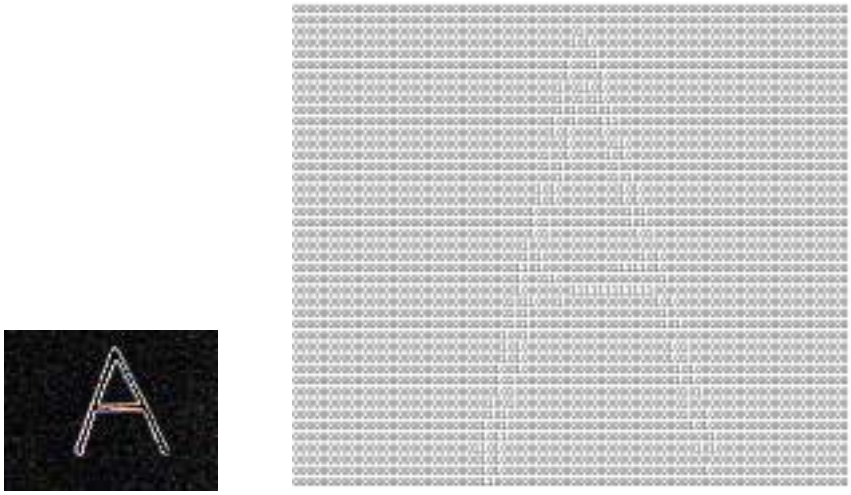


Figure 7: Raw data output of handwritten character

Once the raw data for the edge image has been produced it can be passed onto the perception/recognition stage. It is at this stage that the computer vision system attempts to make sense of the data. There are many different techniques that can be used in order to make sense of this raw data and the choice of technique will depend on the requirements of the computer vision system being developed. For the purpose of this paper we will focus on one such technique that is particularly adept at recognising patterns within the raw data.

4 Perception/Recognition

Since the 1980's it has been has been widely accepted that neural networks are excellent at solving pattern recognition problems. Although a full description of neural networks is beyond the scope of this paper some concepts and terminology must be introduced that will allow for a description of how they can be used in computer vision projects. Neural networks are vastly simplified computational models of the biological human neuron [6]. Each biological neuron is made up of three main parts, the axon, dendrites and the cell body or soma. The dendrites detect electrical impulses from neighbouring cells a pass these impulses

into the cell body. The neuron is said to 'fire' if the sum of arriving impulses is above a certain threshold value. When the neuron fires it sends another electrical impulse down its axon, this electrical impulse is then detected by the dendrites of other neighbouring cells. In 1958 Frank Rosenblatt developed a simple computation device called the Perceptron that was based on the work carried out by Hebb, McCulloch and Pitts in 1943. Rosenblatt's work still forms the basis of the neural networks that are used today. Figure 8 shows the similarities between a biological neuron and Rosenblatt's Perceptron.

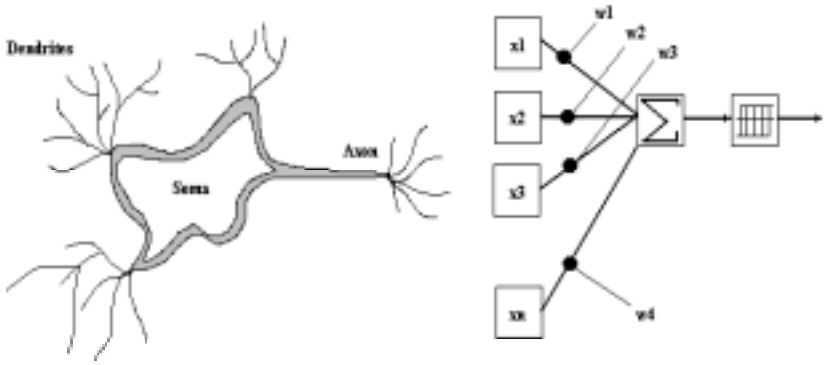


Figure 8: Comparison of biological neuron and Perceptron

Each Perceptron has a number of inputs that are connected to a summation unit via weighted links. The summation unit calculates the weighted input and passes the result to a threshold unit. The threshold unit outputs a 1 if the weighted input is above or equal to some value, otherwise it outputs 0. A learning algorithm is usually used to optimise the weight values associated with each link in order to give the desired result. Complex networks can be built by connecting many of these simple computational structures together, however, implementing these networks in software can be quite difficult and time consuming. The approach taken by many neural network researchers is to develop a software library of classes that can be used and modified over time to speed up the implementation and testing of new types of networks. Computer science students studying at ITB have access to one such library, the library is comprised of a set of Java classes that can be used to implement various kinds of neural networks [7]. The library provides a set of primitive neural network classes that students can use to build their own networks or alternatively, students can use a set of existing network classes such as the BackProp (backpropagation) class to solve problems [8].

4.1 Using the Java Backpropagtion Class

By using the Backprop Java class, students can concentrate on the problem the network is attempting to solve rather then the implementation details of the network itself. Students can

build and train a backpropagation network by constructing a topology and training file and passing these files as parameters to a Backprop object. The Backprop object also provides *learn* and *run* methods that the student can call to operate the network in training mode or in normal mode. Using the network in training mode will cause the Backprop object to read the training file and begin readjusting its weights using the backpropagtion algorithm. Once the network has been trained and its weights have been saved, it can be used in normal mode to solve problems such as pattern recognition. Although a lot of the more difficult implementation details are taken care of by the neural network Java classes, the student is still required to build training files and decide on an appropriate topology for the network. Figure 9 shows the contents of a typical topology and training file.

```
// Topology file                          // Training file
                                          0 a1.trn 1 0 0 0
0.45  // Learning rate parameter          1 a2.trn 1 0 0 0
0.9   // Momentum term                    2 b1.trn 0 1 0 0
0.1   // Tolerance                        3 b2.trn 0 1 0 0
3     // Number of network layers         4 c1.trn 0 0 1 0
4800  // Number of input nodes             5 c2.trn 0 0 1 0
100   // Number middle layers nodes       6 d1.trn 0 0 0 1
3     // Number of output nodes           7 d2.trn 0 0 0 1
```

Figure 9: Topology and training files used by Backprop class

The topology file allows the student to set parameters associated with the backpropagation learning algorithm such as the learning rate and momentum term. The student can also define the number of layers that the network contains and the number of units on each layer. These values can be adjusted after each training session to improve the overall performance of the network. The construction of the training file is less straightforward and will depend heavily on the problem the network is required to solve. The training file shown in figure 9 is used to present the network with some positive examples of alphabetical characters. Each line of the training file must start with a pattern number followed by the name of a file that contains a raw data representation of the letter to be learned (see figure 7), finally each line of training file must end with the desired output for that pattern. The desired output data informs the network which output unit should fire for each pattern. So for example, if the network is presented with an A, the first output node should fire, if the network is presented with a B, the second output unit should fire, and so on. Once the topology and training file have been constructed the student can use the Backprop class by constructing a Backprop object and passing the topology and training file as parameters. The code listing below shows how this can be done.

```
// Build the network
Backprop bp = new Backprop("topology.txt", "alphabet.trn");
bp.createNetwork();

// Train the network and save its weights
bp.learn()
bp.saveWeights("weights.dta");

// Load the saved weights and a new pattern to test the network
bp.loadWeights("weights.dta");
bp.resetInputPatterns("test.trn", 1);
bp.run();
```

Listing 4: BP network construction, training and running

Using the Backprop class as part of a computer vision project requires the student to write two separate programs, a training program and an application program. The training program must acquire images that can be presented to the network during the training phase, and the application program must use the weights saved during the training phase to recognize patterns that the network has not seen before. The software libraries introduced in sections 2 and 3 of this paper can be applied to both the training program and the application program.

5 Computer Vision Application

This section introduces a demonstration application that was developed using the software libraries introduced in sections 2, 3, and 4. The application is capable of recognising three categories of everyday objects, Pens, Apples and Phones. A flowchart of the program execution can be seen in figure 10.

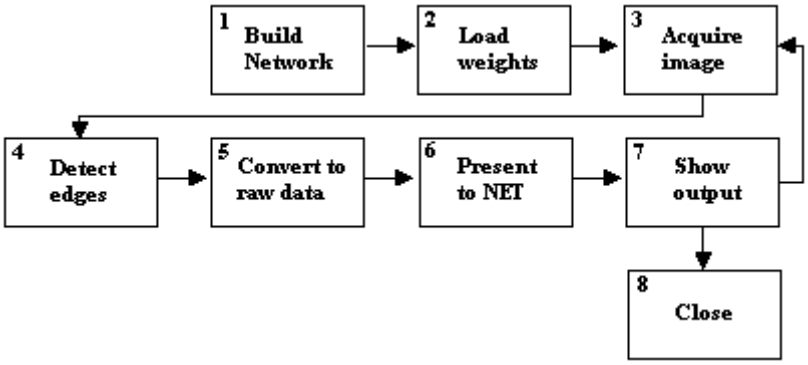


Figure 10: Flowchart showing program execution

Stage 1: The program begins by constructing the neural network that will be used during stage 6. In this example a 3-layer network with 4800 inputs, 150 middle units and 3 outputs is used. Each of the 4800 inputs represents one pixel of the input pattern. Each of the 3 output

units represents the category that the input image belongs to. Figure 11 shows the topology of the network used.

Stage 2: This stage of the program execution loads the weight values of a previous backpropagation training session. In this example the network was trained using 39 patterns containing 12 examples of pens,13 examples of apples and 14 examples of phones.

Stage 3: This stage uses the Java TWIAN API to acquire an input image from a USB WebCam attached to the system.

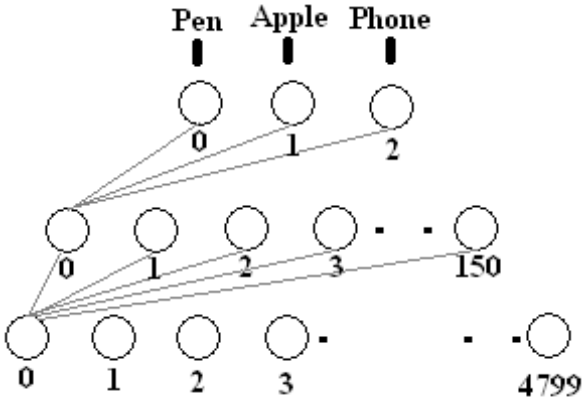


Figure 11: Neural network topology

Stage 4: This stage extracts an edge image from the input image using the Sobel masks introduced in section 3 of this paper.

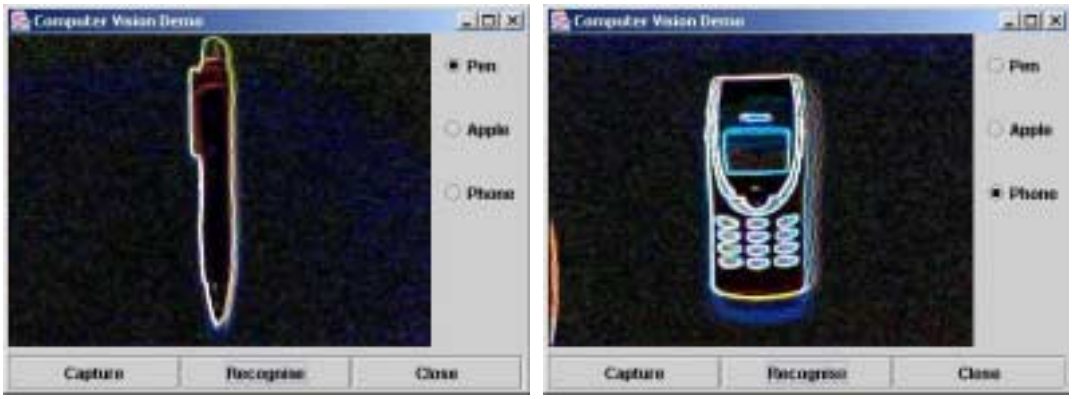
Stage 5: Once an edge image has been produced it is converted into raw data using the thresholding operator discussed in section 3. Once the raw data has been produced it is written to a file called 'edge.trn'. This file will then be used as input to the network.

Stage 6: In this stage the information contained in the edge.trn file is presented to the network. As the network is operating in normal mode it only has to deal with one image at a time. The network takes each input value and loads the appropriate input unit with that value. It then feeds the input values forward and set the values on the output nodes.

Stage 7: Once the network has completed its feed-forward cycle the program checks the value of each output node to see which one is the highest. For example, if output node 0 has the highest values the network has recognised the input image as a pen, if the second output node has the highest value then the network has recognised the input image as an apple, and so on. Once the highest values have been determined the program notifies the user of the

result. From here the user can choose to quit the application or test the network again by acquiring another image.

Figure 12 shows the application's user interface. The user interface consists of a panel to display the edge image, a panel to show the network output and a panel that contains buttons that allow the user to acquire an image, recognise the current image and close the application. The user can test the network by clicking the 'Capture' button to acquire an image and then clicking the 'Recognize' button to present the pattern to the network. When the user clicks the 'Recognize' button, the network output will be displayed by switching on the appropriate radio button.



Input correctly identified as pen Input correctly identified as phone

Figure 12: Application user interface

6 Conclusion

The application discussed in section 5 of this paper integrates the functionality of the software libraries introduced in sections 2, 3, and 4. The demo application follows the decomposition of work discussed in section 1 and applies the appropriate software tools at each stage. Although the software libraries used cut down on the amount of coding required, integration of each stage of the computer vision model is left up to the student. The demo application leaves a much room for improvement and is by no means a fully blown computer vision application. For example, the range of objects the application can recognise is limited and only very basic image processing is carried out during the pre-processing stage. The intention here is that it is up to the student to improve the system.

A list of possible areas where the student could improve the system is given below.

- Adjustment of WebCam parameters
- Colour analysis
- Input image orientation
- Dealing with multiple images
- Extraction of higher order data from images
- Better raw data conversion
- Optimisation of learning parameters
- More complete training sets
- Implementation of more sophisticated neural networks
- Experimentation with other problem domains

Overall the choice of software libraries has been successful, the libraries deliver the functionality required for computer vision projects and are easy to use. One possible improvement as regards the choice of libraries would be to use the JMF (Java Media Framework) for image acquisition as this library provides a pure Java interface to devices such as WebCams and can be used to capture time-based data, i.e. Digital Video.

References

[1] Dean, Allen & Aloimonds (1995). T. Dean, J. Allen, & Y. Aloimonds, Artificial Intelligence: Theory and Practice, *Addison Wesley*, 9.3:415-417.

[2] TWAIN Working Group (2000). TWAIN Working Group, TWAIN Specification Version 1.9, *TWAIN Group*, 1:1-3.

[3] TWAIN Working Group (2000). TWAIN Working Group, TWAIN Specification Version 1.9, *TWAIN Group*, 9:340-498.

[4] Parker (1997). J. R. Parker, Algorithms for Image Processing and Computer Vision, *John Wiley & Sons, Inc*, 1:1-7.

[5] Marr, Hildreth (1980). D. Marr, E. Hildreth, Theory of Edge Detection, *Proceedings of the Royal Society of London*, Series B. Vol. 207:187-217.

[6] Arbib (1995). M. A. Arbib, Introducing the Neuron, *Handbook of Brain Theory. MIT Press*, Part I.:4-11.

[7] Sheridan (2000). S. Sheridan, Non-Deterministic Processing in Neural Networks, *ITB Journal, Issue 2*:4-20.

[8] Werbos (1995). P. Werbos, Backpropagation: Basics and New Developments, *Handbook of Brain Theory. MIT Press*, Part III.:134-139.

Longer Than A Telephone Wire - Voice Firewalls To Counter Ubiquitous Lie Detection

Carl Reynolds[1], Matt Smith[2], Mark Woodman[1]

[1] *School of Computing Science,*
Middlesex University, London UK
[2] *School of Informatics and Engineering,*
Institute of Technology at Blanchardstown, Dublin 15
Contact email: c.reynolds@mdx.ac.uk

Abstract

Mobile computing and communication devices are open to surreptitious privacy attacks using emotion detection techniques; largely utilising work carried out in the area of voice stress analysis (VSA). This paper extends some work in the area of removing emotion cues in the voice, specifically focusing on lie detection and presents the results of a pilot study indicating that the use of mobile phones in situations of stress is common and that awareness of VSA is low. Existing strategies for the removal or modification of emotion cues, based on models of synthesis are considered and weaknesses are identified.

Keywords

Lie detection, privacy, mobile devices, voice firewall, voice stress analysis, emotion detection, speech processing.

1. Introduction

Invasions of privacy and security in mobile telecomm equipment are often assumed to come from third parties; this could for example involve the use of scanning devices, line-tapping or eavesdropping. It is possible that the receiving party in a voice communication session may also carry out a voice stress analysis (VSA) of a speaker, thus providing an invasion of privacy. A variety of "emotion detecting" VSA devices are readily available with manufacturers and distributors making considerable claims as to their efficacy, despite the many studies casting doubt on the reliability of VSA devices, even within the restricted context of lie detection. Meyerhoff, Saviolakis, Koening, & Yurick, (2001), and Hollien,

Geison & Hicks (1987) provide evidence of this; most researchers suggesting results that provide little better than chance at accurate emotion detection.

There is sufficient research in emotion detection through voice analysis to suggest that although we may not clearly state the accuracy of these techniques in a variety of situations, the sharing of common strategies over many research groups is still indicative of a reliable body of knowledge in this area.

Emotion detection through audio analysis is not restricted to speech, according to a local newspaper, a device "Why Cry" has just become available. This analyses a baby's crying and has a claimed 98% reliability in detecting if a baby is bored, stressed, uncomfortable, hungry or tired (Miranda, 2002). These unsubstantiated claims indicate a popular interest in this area, and the use of lie and emotion detectors is becoming common in popular television shows.

The area of VSA and thus lie detection is largely informed by research into prosody, a term that describes pitch contours, rhythm and chunking (grouping of syllables). There has been considerable work in the investigation of prosody changes in a variety of emotional states and prosody analysis has been extended to include detailed analysis of the frequencies and noise present in speech. Properties that are of particular interest include the fundamental frequency (F0), which is a measure of the pitch and changes in energy in different frequency bands.

Scherer, Johnstone and Banziger (1998) have identified changes in speaker state, a collective term for different types of stress, emotional outlook and attitude and are looking to develop more robust strategies for speaker verification. In their work they have collated a substantial amount of information about the ways in which emotional, cognitive and physiological stress may modify measurable properties of speech. There work suggests that the movement and range of F0 is important together with its energy. This is supported by other work in this area, such as Li & Zhao (1998).

Voice Stress Analysis uses various cues in the speech signal to ascertain the emotional state of the speaker; there has been considerable research in this area, summarised by Scherer (1995) and presented in a simplified form in Table 1. This paper only considers the 7 factors in Table 1 and five emotional states, although Schuller, Lang & Rigoll (2002) use an 18 dimensional feature vector in order to distinguish between seven emotional states. The neutral user state is assumed in our simplified model, although in practice it would be derived in a calibration period during the operation of a VSA device and as such may not actually be

neutral. Many VSA devices only consider stress and neutral state as important (in use as lie detectors). Some research in VSA fails to consider a neutral state and this could lead to some inaccuracies in findings, however as stated previously, the consensus is that the movement and energy of F0 is a key indicator of speaker state, this holds true whether or not a neutral state is used to calibrate results.

It is important to note that this calibration to determine a neutral state is very important. It might be used by an informed speaker as an opportunity to voice the "big" lie at the start of the conversation, and thus to confuse the lie detector's calibration system. In such a situation it would be possible to take a sample of the same speech from the end section of the speech (preferably during a known "truth statement") and to stitch this to the beginning. This allows a reliable calibration procedure and the "lie to be exposed", although not in real time.

Change to Speech	Anger	Fear	Sadness	Joy
Mean F0	Increase	Increase and higher	Decrease	Increase
F0 range	Increase	Increase	Decrease	Increase
F0 variability	Increase			Increase
F0 contour Direction	Downward		Downward	
Mean intensity	Increase		Decrease	Increase
High frequency energy	Increase	Increase		Increase
Articulation rate	Increase	Increase	Decrease	Increase

Table 1: Summary of existing work (from Scherer 1995).

Reynolds, Smith & Woodman (2002) described a number of strategies for blocking VSA that have been summarised in Table 2.

Strategy	**Process**	**Advantages**	**Disadvantages**
Subtractive	Complex waveforms are filtered and amplified to modify the waveform, by subtracting harmonics.	May be subtle, possible to run in real time.	Does not modify changes in F0.
Additive	Simple waveforms are combined to make more complex waveforms.	Easy to implement in real time.	Easily detected, but only modifies Jitter.
Spectral Modeling	Use of transforms to allow manipulation of audio in the frequency domain. In addition to tone control it encompasses audio effects such as pitch tracking and shifting.	Could be undetectable and allow very fine control over a VSA response.	Considerable processing power required.
Avatar	Phones are stored in a look up table and referenced as required giving a substitution for neutral speech.	A complete firewall.	Easily detected and intelligibility only adequate.

Table 2: Strategies for Blocking VSA.

There have been many other studies investigating emotion in speech and some recent approaches, such as that by Schuller, Lang & Rigoll (2002) analyse the content for semantic clues, such as key words, verbosity and non-verbal utterances in addition to consideration of the psychophysiological data. The context of their work is in the area of interaction design and adaptable interfaces, although it may yet play a role in other forms of emotion detection. The work uses acted emotions and this might produce very misleading results, although they publish confusion tables, supporting Reynolds, Smith & Woodman's (2002) observations that "*very different emotions, might elicit the same stress response for a given indicator*". There is evidence that types of stress such as emotional, physiological and cognitive produce different changes in formants or harmonic content in the vowel sounds, with some change being dependent on the speaker's coping strategy (Tolkmitt & Scherer, 1986). Identifying such a strategy in the analysis of small amounts of speech might prove to be difficult.

Many early VSA devices used the presence or absence of micro tremors or jitter (Smith, 1977) to determine stress and anxiety in the speaker. See Figure 1 for a plot of jitter. These reduce with a corresponding increase in anxiety; these manifest themselves as a 3 – 10Hz frequency modulation of the speech signal and may increase in frequency. Prevaricator is an example of a software lie detector that takes this approach to anxiety analysis (Prescott, 1999).

To test the results of emotion research, it is necessary to have a corpus of test data. Many such corpi exist (SUSAS) although the material contained in them may be acted or elicited speech rather than real emotional speech. For the purposes of accurately testing VSA it would seem that real lies would be most appropriate. These are hard to obtain and there may be many questions about the ethical and legal issues. For this work in blocking VSA the accuracy of the original test data is not considered important because we need to provide consistently neutral or predetermined results for any speech input, whether acted or a genuine lie. However the use of acted emotions must cast some uncertainty on the reliability of VSA, when these are solely used as test data, since the stress detected may not be real.

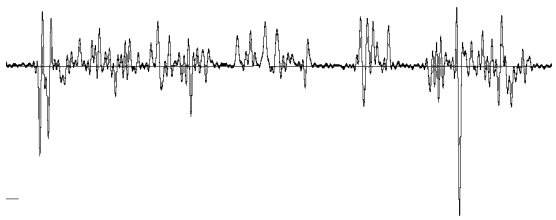


Figure 1: Amplitude/time plot of jitter (normalised).

This work extends strategies for blocking emotional cues and may have an impact on intelligibility, especially in the presence of competing audio streams. It could be claimed that the audio cues that allow auditory scene analysis to take place include the presence of jitter or micro tremors and the movement of F0. This seems likely when considering the work of Bregman (1990) and plays a significant role in our understanding of the cocktail party effect (Cherry 1953 and Arons 1992). In considering strategies for the removal of VSA cues it may be important to consider the retention of as much prosody information as possible, especially where monophonic devices with limited audio bandwidth are concerned. Removal of VSA cues may also lead to the removal of the cues that aid formation of audio streams, this may be of importance where phone conferencing systems are used. No work has yet been carried out to determine the impact of voice stress cue modification on the ability to form coherent primitive audio streams.

In addition to this, removal of cues for emotion may lead to confusion over meaning. This may be noticeable in the case of irony, sarcasm and when telling jokes. With many chatrooms using emoticons, this concept could be extended to mobile voice communications with the addition of specific aural clues.

2. Developing the Strategies for "Voice Firewalls"

The previously considered strategies for preventing the surreptitious use of lie detectors have been refined and the work carried out as follows:

Subtractive

This approach is largely effective against the use of Jitter (see Figure 1) or micro tremors as a means of determining the veracity of a speaker. Setting a high pass filter at 20Hz has no noticeable effect on speech quality and no detection has been indicated in listener tests (Jitter is transmitted by restricted bandwidth devices such as mobile phones). Such filtering easily prevents Prevaricator from producing a high stress result. It was suggested that a subtractive approach be used for adjusting energy in frequency bands, however if a "Voice Firewall" is to be developed against more sophisticated detectors that analyse prosodic cues, then the processing would be better carried out in the frequency domain.

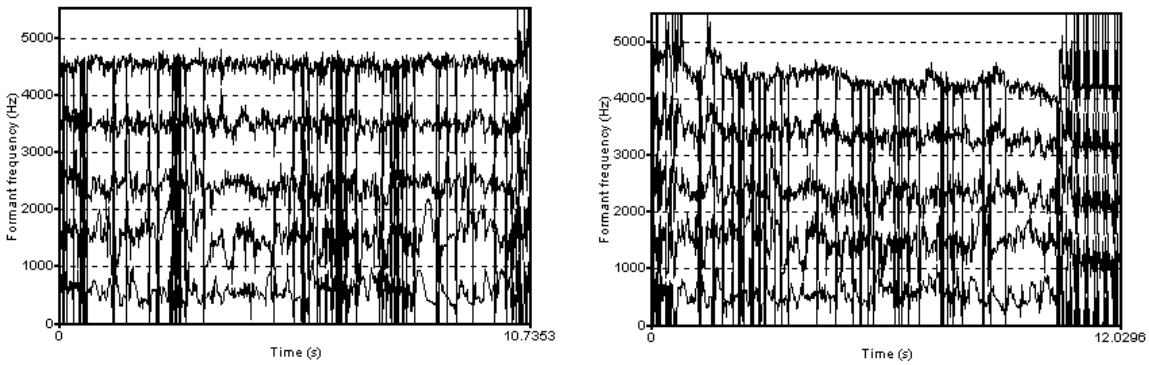
Additive

This is best used with the subtractive approach and replaces the Jitter pattern in the voice signal with a constructed or prerecorded one containing false emotional cues. It can be used to fool simple detectors but is less effective against an emotion detection approach that considers variation in the mean F0 of the voice. The jitter signal remains inaudible, even when normalised.

Spectral Modeling

The advantage of a spectral modeling approach is that it utilises the frequency domain. This enables the use of complex filters and formant tracking and modification. Simple pitch correction techniques can be applied; these are able to shift all formants including F0 to enable some of the natural qualities of the voice to be maintained. It is possible to make small changes in real time, although the latency of such an approach has not been calculated. Figure

2 shows the change to formants when pitch changes in real time are applied. This example did not use the original speech parameters to control the pitch shift; this was predetermined by settings.



(a) Original Formants **(b) Pitch Shifted Formants**

Figure 2: Formant change with real time pitch changes.

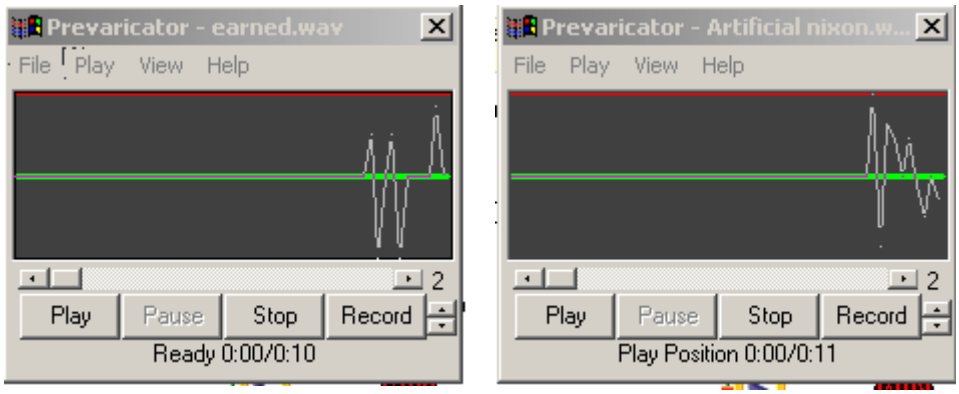
Avatar

The avatar method makes use of a virtual speaker or voice avatar that copies the original. Such a speaker could be a neutral clone of the speaker with no voice stress cues, another person, or a fictional character. This strategy utilises technology that exists in speech recognition and in text to speech software. We can utilise software to convert the user's voice to text and then read this text with an artificial voice. This provides a complete voice firewall solution. It includes the stage of providing a textual output, which involves a requirement to ensure that the text makes sense and that words are correctly spelled. This would not be required for a voice firewall, because the listener performs the required cognitive processing but does not receive the voice stress cues thus making VSA futile. In the proposed avatar approach the spoken words are analysed and parsed to extract the phoneme content (phonemes being basic sounds that form the basis for words). The identified phonemes can reference a look-up table that stores a digital copy of a voice stress neutral equivalent of the phoneme which can be sent to the output. A phoneme palette could be stored in firmware.

It could still be possible to extract some prosodic elements such as chunking from the speech data, to assist in increasing intelligibility.

Many text to speech software packages now have more realistic virtual speakers, although earlier criticism of such systems is that the results are mechanical (Hardman, Sasse, Handley

& Watson, 1995) and that intelligibility is only adequate. The avatar strategy was tested by using a text to speech package and capturing the output. This was then analysed by two lie/emotion detectors. The results illustrated in Figure 3 are for the Nixon "I am not a crook" speech.



(a) Read by Nixon **(b) Read by virtual speaker**

Figure 3: Nixon's speech

The results indicate that the speaker is under stress, and therefore the virtual speaker's voice characteristics in this instance are unexpected. The development of a neutral phoneme database is therefore indicated as important in the context of this work.

3. Analysis of the proposed strategies

In analysing the strategies we have come to the conclusion that spectral manipulation provides an effective form of voice firewall whilst retaining the original speakers voice characteristics. Also the avatar approach offers complete isolation from the psychophysiological cues that may be present in speech. The subtractive and additive approaches have been evaluated with success against simple lie detectors and these can be used surreptitiously. These approaches are not robust enough for blocking emotion detection where sophisticated analysis, such as F0 tracking takes place. The most isolating approach is the avatar, however this needs to be developed to ensure that the phoneme database contains neutral phonemes. These could be based on the speaker's own voice or the voice of another and may have implications for voice recognition.

4. Evidence of Requirement

During 2002 a pilot survey of mobile phone use was conducted of 32 London commuters. The following results were obtained:

1. Do you use a mobile phone or other mobile voice communication device for business purposes?

   YES **14** NO **18** *(2 had no phone)

2. Do you ever use your mobile device to transmit sensitive information or decisions?

   YES **5** NO **9**

3. Are you aware of any security or privacy risks when transmitting sensitive or confidential information?

   YES **7** NO **7**

4. Have you ever been in stressful situations when transmitting such information?

   YES **11** NO **3**

Although a relatively small, informal survey, making no claims for statistical significance, these results indicate that it is not unreasonable for us to suggest that mobile phones are:

- Used for business purposes, and
- Used to transmit sensitive information.

It is also reasonable to suggest that many users would be unaware of security or privacy issues and that the majority were likely to use their phone in stressful situations. Those who were aware of privacy risks considered eavesdropping as the main risk followed by scanning. No users considered the use of emotion detection. Two of the users who claimed to have never used a mobile phone under stress admitted to answering their phone while driving and did not consider this to be stressful.

5. Conclusions

Four strategies for blocking the use of emotion/lie detectors have been reduced to two, with the subtractive and additive approaches only being effective against the most primitive of stress/lie detectors and these may be considered a subset of spectral modeling.

The techniques developed, although still not fully implemented in a stand–alone form are capable of blocking lie detectors, but there are trade-offs on intelligibility, as yet not fully measured.

The following conclusions can be made:

- The avatar strategy may not be able to use an "off the shelf" phone database, as these may not be emotion neutral. This suggests that future work should include the generation of emotion neutral phoneme databases.
- The spectral modeling strategy could allow some elements of prosody to remain, although the extent to which the intelligibility of the speech remains unaffected has not been measured.
- The proposed strategies may be implemented by modifying existing algorithms developed from models of synthesis.
- It is possible to confuse voice stress analysis based lie detectors in many ways. This is of value not only with the need to retain privacy, but also because these detectors often do not appear to have the accuracy claimed or that the accuracy claimed is given for specific experimental conditions.
- Mobile communications devices do catch their users unguarded, as they often are involved in other tasks that are not related to the issues involved in the communication itself. The pilot survey carried out illustrates that mobile users are answering their phones in stress situations, even when not recognised as such. It also illustrated a lack of awareness of privacy issues, and use of mobile communication equipment as a business communication tool. A more extensive survey is required in order to understand the impact that psychophysiological tools might have on the way we disseminate information in an increasingly "mobile" world.

6. References

B. Arons (1992). A review of the cocktail party effect. *Journal of the American Voice I/O Society*, 12 (July)

A. S. Bregman (1990). *Auditory Scene Analysis: The Perceptual Organisation of Sound.* The MIT Press.

E. C. Cherry (1953). *Some experiments on the recognition of speech with one and with two ears*. Journal of the Acoustical Society of America, *25, 975-979*

V. Hardman, A. Sasse, M. Handley & M. Watson (1995) *Reliable Audio for Use over the Internet*. Presented at INET '95

H. Hollien, L. Geison & J. W. Hicks Jr. (1987) Voice Stress Evaluators and Lie Detection, *Journal of Forensic Science* 8703 Vol 32 Iss2 405-418

Y. Li & Y. Zhao (1998) Recognising Emotions in Speech Using Short-term and Long-term Features. Proceedings of the ICSLP. pp. 2255-2558.

J. L. Meyerhoff, G. A. Saviolakis, M. L. Koening & D. L. Yurick (2001) DoDPI Research Division Staff, *Physiological and biochemical measures of stress compared to voice stress analysis using*

the computer voice stress analyzer (CVSA). (Report No. DoDPI01-R- 0001). Fort Jackson, SC: Department of Defense Polygraph Institute, & Washington, DC: Walter Reed Army Institute of Research.

E. Miranda (2002) So that's what the baby's trying to say when it cries. *Evening Standard* Monday 14 October 2002

M. C. Prescott (1999) *Prevaricator* Software v1.04 Freeware.

C. Reynolds, M. Smith & M. Woodman (2002) "*Your Nose is…"*. In Proceedings of SCI 2002, the Sixth World Mutliconference on Systemics, Cybernetics and Informatics, Orlando Florida, USA, July 2002.

K. R. Scherer, T. Johnstone & T. Bänziger (1998) (1998, October*). Automatic verification of emotionally stressed speakers: The problem of individual differences*. Paper presented at SPECOM'98, International Workshop on speech and Computers, St. Petersburg, Russia. Geneva Studies in Emotion and Communication, 12(1).

K. R. Scherer (1995) "Expression of emotion in voice and music", *J. Voice*, 9(3), 1995, 235-248.

B. Schuller, M. Lang & G. Rigoll (2002) *Automatic Emotion Recognition by the Speech Signal*, Institute for Human-Machine-Communication, Technical University of Munich 80290 Munich, Germany, presented at SCI 2002. CD-ROM conference proceedings.

G. A. Smith (1977) Voice analysis for the measurement of anxiety. *British Journal of Medical Psychology*, 50, 367-373.

SUSAS (2002) SUSAS *Speech Under Simulated and Actual Stress*. Web reference located at: http://www.ldc.upenn.edu/ldc/news/release/SUSAS.html

F. Tolkmitt & K. R. Scherer (1986) Effects of experimentally induced stress on vocal parameters. *Journal of Experimental Psychology: Human Perception and Performance,* 12, 302-313.

An XML Framework for the Structured Data Exchange Between Medical Devices

Ulrich Neuhaus [+], Paul Walsh [#]

Cork Institute of Technology, Rossa Avenue, Bishopstown, Cork, Ireland

[+] ulrich.neuhaus@e-healthcare.de, [#] pwalsh@cit.ie

Klaus W. Wente

Fachhochschule Darmstadt, Schoefferstr. 6. Darmstadt, Germany

k.wente@fbi.fh-darmstadt.de

Abstract

This paper describes an XML framework for the exchange of data between medical devices and software systems. XML technologies offer an extensible data format, which is easy to both create and process and simplifies the exchange of data. Different XML standards are defined within the framework in order to provide a practical solution for both device manufacturers and software developers. The structure and benefits of XML documents are discussed and an XML framework for ophthalmology is described.

Introduction

This paper describes an XML framework for the data exchange between medical devices in the area of ophthalmology. In ophthalmology, an examination contains measurements and maybe images of a patient's eyes. Even for one examination, different devices can be used and the collected information about the patient's eyes has then to be interpreted by a physician. Most of the measurements are done by devices that are connected to a software system. The physician should then be able to get all required information from an electronic patient record system, and use this information to treat the patient accordingly.

In the area of data exchange between medical software systems a wide variety of different standards for different purposes are available, e.g. Health Level 7 [1] for patient information, billing and accounting or DICOM [2] for digital imaging and structured reports. However the communication between medical devices and software systems is an area where virtually every vendor defines and uses their own specific data formats. For example, examination data is stored in a fixed length string, where all information has a known length and position in the string. Another variant on this scheme is a string separated by a control character such as a comma or semicolon, where the length of values is variable but the order is always fixed.

Changes and extensions to both formats require a customisation of the application interface and occasionally the device interface. Because of these reasons the existing software systems implement various interfaces to all or most available devices to connect them to the application. Those implementations work properly, but with every new or changed device, the existing program code gets larger and the maintenance for those interfaces gets expensive. At this point it would be helpful to have one data format which has to be processed.

In this paper, different XML technologies are used for describing the structure of data that is communicated between medical devices and software applications. The use of XML provides both device manufacturers and software developers with a powerful solution, which could solve the existing problems and make data exchange easier and cheaper. Moreover, XML is a standardized format and there are a lot of libraries for creating, accessing and parsing XML documents very quickly, e.g. MSXML Parser [3] or SAX [4]. For the software developer it is easier to process an XML document, because the processing of the received document is always the same. The developer only needs to know which data elements are relevant and can frequently reuse the same code for processing that data. The vendor gets a data format, which is easy to extend for future requirements. The use of a *standardized* format is also a useful selling point. By using a standard format, the idea of connecting a device easily to a software system (like plug and play) becomes possible.

As there are a wide range of medical devices available that cater for every medical field, the system described here is focused only on a subset of devices in the area of ophthalmology. Indeed a lot of standardization work has been done in the area of ophthalmology and the readiness of manufacturers for participating in such a framework is encouraging. If a usable framework is developed and verified, it could be adapted to other medical areas.

XML Technologies.

The XML language [5] makes it possible to structure and process almost any kind of information. All information is stored between markup tags or in attributes. An XML document contains markups and data. A markup is used for describing the document's layout and logical structure, while the data is the main information described in the document. An XML document is normally human readable.

An XML document is only valid if the document is well formed. This means, that all information is stored between markups and every opened tag is closed. This function is offered by an XML schema. An XML schema [6] is a document that formally describes an XML document and can be used to prove if the information inside markups is valid.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <Autorefractor PatientGuid="471143" ExaminationDate="2002-08-01"
3   AFType="manual" DocumentID="898890">
4           <AFLeft>
5                   <Geometry>
6                           <Sphere>120</Sphere>
7                           <Cylinder>100</Cylinder>
8                           <Axis>90</Axis>
9                   </Geometry>
10          </AFLeft>
11          <AFRight>
12                  <Geometry>
13                          <Sphere>100</Sphere>
14                          <Cylinder>140</Cylinder>
15                          <Axis>70</Axis>
16                  </Geometry>
17          </AFRight>
18          <Remarks>some remarks</Remarks>
19  </Autorefractor>
```

Figure 1: XML document for an auto refractor

Figure 1 shows a valid XML document. The tag "Autorefractor"[1] is the root element and contains four attributes (PatientGuid, ExaminationDate, AFType and the DocumentID) and three child elements. These elements store the measurement values "AFLeft" and "AFRight" from the examination and some free text "Remarks". Every XML Document can be displayed as a tree-structure with one root and several child elements (see Figure 2). It is not possible for a document to have two root elements; if this happens the document is not valid.

Within the XML Schema, the structure of an XML document is described. For each element and attribute the name and the data type are defined. The schema also includes information about:

- optional and required attributes and elements;
- the order and the occurrence of elements;
- the range of values for attributes and elements.

[1] An Autorefractor is a device for measuring the refractive power of an eye.

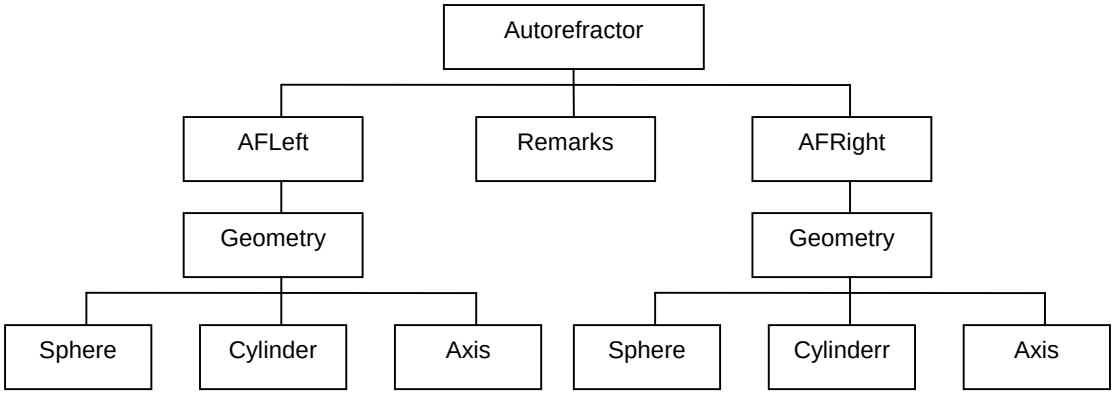


Figure 2: XML tree-structure

Schemas are used in two different ways. The creator of a XML document uses the schema to retrieve all information necessary for creating a valid document. The software developer or the application uses the schema to validate the document against it.

A language for accessing elements in an XML document is also specified. Simple access to any element in the document is offered by XPath [7]. With XPath a query language is available that is similar to the functionality of SQL for relational databases, but is much easier to use. Every element and attribute in the document is accessible by a path. In Figure 3, the first query gets the value of the element sphere; the second query retrieves the value from the attribute PatientGuid.

/Autorefractor/AFLeft/Geoemtry/Sphere

/Autorefractor/@PatientGuid

Figure 3: Simple XPath Queries

Schema Definition for Medical Devices

Initially all information must be defined using XML schemas. Four different types of XML schemas, shown in Figure 4, were developed.

Different approaches were used for defining these schemas. The examination schemas were defined by using the information about the data format from the specifications of the respective device, e.g. measurement and range of values. The examination schema for a perimeter[2] was also developed along with the device manufacturers Interzeag [8] and Oculus

[2] A perimeter is a device for measuring the visual field of an eye.

[9]. After collecting the required information, it was converted to a tree-structure to meet the XML requirements.

The examination document is defined by the examination schema. This document stores all data acquired during the examination. The data is divided into the results for the right and left eye. The schema includes three schemas for the patient, the device and the self-defined data types. These additional schemas are optional. When the examination document is created, it may also contain a sub document defined by the patient schema and one defined by the device schema.

In this scope, the examination document contains only the measurements from one device. If a physician is performing several examinations on a patient, the results from all used devices are stored in a new examination document.

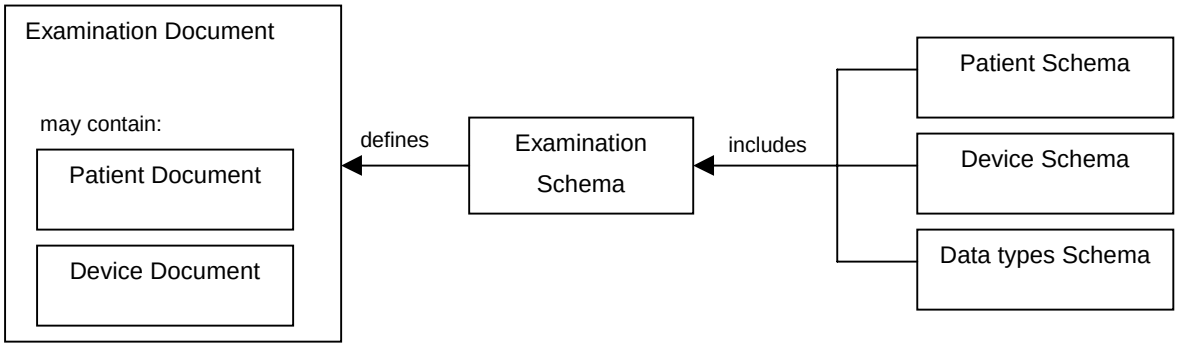


Figure 4: Schema types

- **Patient Schema**

  In the patient schema, personal and contact information about a person is stored, e.g. patient ID, name, and address. Most of the elements in this schema are optional, because not all information is needed for an examination. Only the patient ID is required for assigning the examination results to the correct patient in an application. The patient schema is referenced in every examination schema as optional data.

- **Device Schema**

  The device schema includes all relevant information about the respective device, e.g. device type, software and firmware revisions. The device schema is also linked to every examination schema as optional data.

- **Examination Schema**

  A schema for every device is developed. In this schema all possible values and elements that could be transferred from the device to the application are defined. Schemas are available for the most recent ophthalmologic devices, e.g. auto refractor. A special

section is defined in every schema for vendor specific extensions. This takes advantage of the extensibility of XML. All data stored in this section won't be validated, because the structure of those elements are not defined in the schema and so are unknown to it.

- **Data type Schema**

  An additional schema is developed for elements, which are used in several examination schemas, e.g. the type geometry is composed of three elements: axis, sphere and cylinder.

Figure 5 shows the XML Schema for an Autorefractor. In line 7 to 9 a prefix is defined for each of the three included schemas. The inclusion is done in line 10 to 12. Every element is composed of a name and a data type. It is possible to use simple data types (e.g. float or string), data types from other schemas (e.g. line 13, the definition of an element from the type "DeviceInformation") and to define complex data types composed of simple or other complex types (e.g. line 18 and 19, the "AutorefractorDataset" contains the elements "Geometry" and "VisualAcuity").

In line 35 the root element is defined and all the elements, which are possible as child elements, are listed. Also the attributes are defined and for the attribute "AFType", the potential values are defined (line 54 to 56).

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <xsd:schema targetNamespace=
3   "http://schemas.ectalk.org/xeot/1.0/base/autorefractor"
4   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
5   xmlns="http://schemas.ectalk.org/xeot/1.0/base/autorefractor"
6   xmlns:Autorefractor="Autorefractor"
7   xmlns:ECTALK="http://schemas.ectalk.org/xeot/1.0/base/ectalkdatatypes"
8   xmlns:DI="http://schemas.ectalk.org/xeot/1.0/base/deviceinformation"
9   xmlns:PATIENT="http://schemas.ectalk.org/xeot/1.0/base/patient">
10          <xsd:import namespace="http://schemas.ectalk.org/xeot/1.0/base/deviceinformation"/>
11          <xsd:import namespace="http://schemas.ectalk.org/xeot/1.0/base/ectalkdatatypes"/>
12          <xsd:import namespace="http://schemas.ectalk.org/xeot/1.0/base/patient"/>
13          <xsd:element name="DeviceInformation" type="DI:DeviceInformation"/>
14          <xsd:element name="Patient" type="PATIENT:PatientInformation"/>
15          <xsd:element name="VisualAcuityBinocular" type="xsd:string"/>
16          <xsd:element name="Remarks" type="xsd:string"/>
17          <xsd:element name="PD" type="xsd:float"/>
18          <xsd:element name="AFRight" type="AutoRefractorDataset"/>
19          <xsd:element name="AFLeft" type="AutoRefractorDataset"/>
20          <xsd:complexType name="ExtType">
21                  <xsd:sequence>
22                  <xsd:any namespace="##any"
23                          processContents="lax" minOccurs="0" maxOccurs="0"/>
24                  </xsd:sequence>
25          </xsd:complexType>
26          <xsd:element name="Ext" type="ExtType"/>
27          <xsd:complexType name="AutoRefractorDataset">
28                  <xsd:sequence>
29                          <xsd:element ref="Geometry" minOccurs="0"/>
```

```
30                          <xsd:element ref="VisualAcuity" minOccurs="0"/>
31                  </xsd:sequence>
32          </xsd:complexType>
33          <xsd:element name="Geometry" type="ECTALK:Geometry"/>
34          <xsd:element name="VisualAcuity" type="ECTALK:VisualAcuityComplex"/>
35          <xsd:element name="Autorefractor">
36                  <xsd:complexType>
37                          <xsd:sequence>
38                                  <xsd:element ref="AFLeft" minOccurs="0"/>
39                                  <xsd:element ref="AFRight" minOccurs="0"/>
40                                  <xsd:element ref="Remarks"/>
41                                  <xsd:element ref="PD" minOccurs="0"/>
42                                  <xsd:element ref="VisualAcuityBinocular" minOccurs="0"/>
43                                  <xsd:element ref="DeviceInformation" minOccurs="0"/>
44                                  <xsd:element ref="PatientInformation" minOccurs="0"/>
45                                  <xsd:element ref="Ext"/>
46                          </xsd:sequence>
47                          <xsd:attribute
48                                  name="PatientGUID" type="xsd:anyURI" use="required"/>
49                          <xsd:attribute
50                                  name="ExaminationDate" type="xsd:date" use="required"/>
51                          <xsd:attribute name="AFType" use="required">
52                                  <xsd:simpleType>
53                                          <xsd:restriction base="xsd:NMTOKEN">
54                                                  <xsd:enumeration value="manual"/>
55                                                  <xsd:enumeration value="auto"/>
56                                                  <xsd:enumeration value="skiascopy"/>
57                                          </xsd:restriction>
58                                  </xsd:simpleType>
59                          </xsd:attribute>
60                          <xsd:attribute
61                                  name="DocumentID" type="xsd:anyURI" use="required"/>
62                  </xsd:complexType>
63          </xsd:element>
64  </xsd:schema>
```

Figure 5: XML Schema for an auto refractor

Data Exchange

The data exchange between a device and an application is shown in Figure 6. In this communication, the respective XML schemas are stored in the device and also in the application.

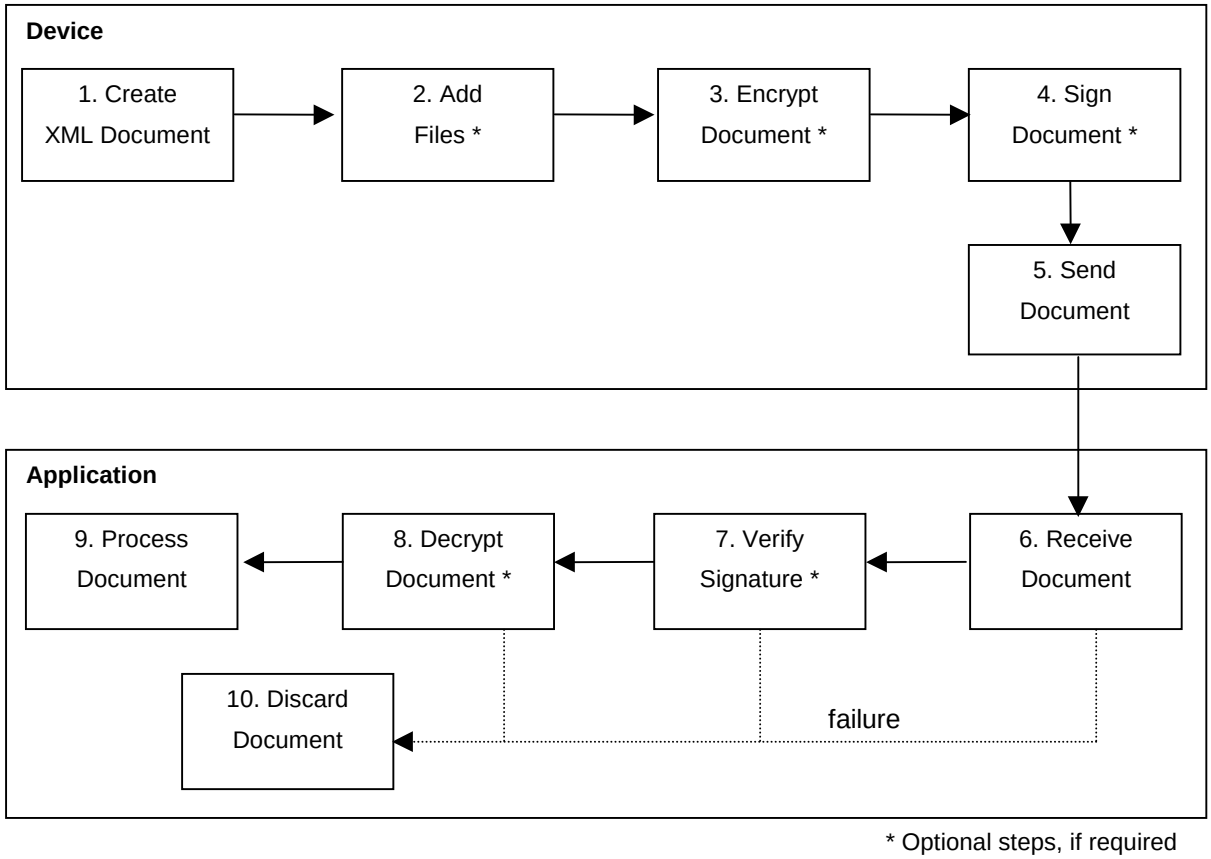


Figure 6: Data exchange

1. Create XML – For every examination a new document is created. At this point, all results are collected from the device application and the document is created, according to the respective schema.

2. Add Files – This step is optional and only required, if binary files (e.g. pictures) should be transferred within the document. The files will be added as string values.

3. Encrypt Document – Because of the sensitivity of all medical information these aspects can become very important, especially when the information is transferred over the Internet or other public networks. Parts or the whole document can be encrypted.

4. Sign Document – In this step the document could be digitally signed to proof authenticity. Steps 3 and 4 are optional. Both steps require a special implementation on the device to allow these security extensions.

5. Send Document – When the document is finally created, it can be sent to the receiving application. Currently the framework does not define how the XML document is to be transferred, e.g. using http. The decision of how this is done is left to the device manufacturer.

6. Receive Document – The document is received by the application and prepared for further processing.

7. Verify Signature – When the document is digitally signed, the signature is verified and if successful the processing is continued.

8. Decrypt Document – If the document is encrypted, the application needs to decrypt it. Steps 7 and 8 also require a special implementation in the application for verifying and decrypting the document. These steps are also optional.

9. *Process Document* – Once all preceding steps are successfully completed, the main processing of the document can begin. The document is verified against the schema, to ensure, that all information in the document is correct and can be processed. It is up to the application, if the information is displayed or stored in a database.

10. Discard Document – If an error occurs in one of the steps, the document is discarded and the examination has to be sent again.

Conclusions

In this paper we have described the structure of an XML framework for data exchange between ophthalmologic medical devices and software systems. The use of XML technologies for structuring has the following advantages:

- it facilitates the use of a standardized data format;
- it is easy to create and process XML data with the existing libraries and additional standards;
- it provides an open data format, which is extensible.

At this stage of the research we have collect all required information about the data output of the devices and defined with this information a base set of XML schemas.

At the moment, one manufacturer (Laser Diagnostic Technologies [10]) is using an earlier version of the XML Schema defined in the framework described in this paper and several manufacturers are interested. The framework contains 15 manufacturer-independent schemas for different medical devices and one vendor specific schema.

In the next phase of research more steps are required. For both, device manufacturer and software developer, a reference implementation is needed. This implementation should show how to create and process XML documents and how to use the XML schemas for validation. Furthermore, the implied security aspects in the data exchange (encrypting, decrypting and signing XML documents) and the use of external resources, which may only be stored in the examination document as an URL, have to be worked out in more detail. For this step, the existing guidelines from the W3C about XML Encryption [11], XML Signature [12] and

XML Linking Language [13] must be incorporated into the framework. Finally, the completed framework has to be verified by manufacturers to ensure a usable solution.

References

[1] Digital Imaging and Communication (DICOM)
http://medical.nema.org/
[2] Health Level 7 (HL-7)
http://www.hl7.org
[3] Microsoft XML Parser 4.0 (MSXML)
http://msdn.microsoft.com/library/en-us/xmlsdk/htm/sdk_intro_6g53.asp
[4] Simple API for XML (SAX)
http://www.saxproject.org/
[51 Extensible Markup Language (XML) 1.0 (Second Edition)
http://www.w3.org/TR/2000/REC-xml-20001006
[6] XML Schema, Part 0, 1 & 2
http://www.w3.org/XML/Schema
[7] XML Path Language (XPath)
http://www.w3.org/TR/1999/REC-xpath-19991116
[8] Interzeag International AG
http://www.octopus.ch
[9] Oculus Optikgeräte GmbH
http://www.oculus.de
[10] Laser Diagnostic Technologies (LDT)
http://www.laserdiagnostic.com
[11] XML Encryption
http://www.w3c.org/Encryption/2001/
[12] XML Signature
http://www.w3c.org/Signature/
[13] XML Linking Language
http://www.w3.org/XML/Linking

Lexical Semantics and Patterns of Causation

Brian Nolan
Institute of Technology Blanchardstown, Dublin

Email: ***brian.nolan@itb.ie***

Abstract

In this paper we provide a brief account of patterns of causation in modern Irish that occur with lexically causative verbs. Three types of causation are found in modern Irish: lexical, periphrastic and morphological. In terms of the relative weightings of each type, the morphological causative is the least productive. Its use appears to be highly constrained to two very specific domains and it is signalled by particular morphological affixes. Lexical causatives are more productive than the morphological causative. By contrast, periphrastic or analytical causatives are highly productive and wide-ranging in their deployment.

A claim of this paper is that an important class of causative constructions are modelled on an underlying schema of caused motion. Within this schema we find that different types of NPs occur to code the end state of the clause, thereby licensing different types of clause structures.

We will demonstrate that there are a number of significant generalisations in the causative constructions that would otherwise be missed, or difficult to find, without the insights inherent in Role and Reference Grammar (RRG) and its logical structure formalism. In particular, we deploy a decompositional representation influenced by RRG to represent the underlying situation types, states of affairs, and events to bring out various uses of the verb cuir 'put' and in so doing we uncover significant evidence to support our contention that motion is a factor in causation along with the eventive primitives of CAUSE, BECOME, INGR and BE. We provide evidence relating to lexically causative verbs in modern Irish whereby they are shown to co-occur with certain prepositional phrases to create periphrastic causative constructions whose semantics is beyond that recorded lexically on the verb.

Our view is that periphrastic causation in modern Irish is concerned with causative motion within an event frame, is sensitive to interpretation as a prototypicality structure and the underlying schemata represent the extensions over this prototype.

1.0 Introduction

In this paper we provide a brief account of elements of causation in modern Irish. Three types of causation are found within modern Irish: lexical, periphrastic and morphological. In terms of the relative weightings of each type, the morphological causative is the least productive. Its use appears to be highly constrained to two very specific domains and it is signalled by particular morphological affixes. Lexical causatives are more productive than the morphological causative. By contrast, periphrastic or analytical causatives are highly productive and wide-ranging in their deployment. A claim of this paper is that an important class of causative constructions are modelled on an underlying schema of caused motion. Within this schema, we find that different of NP types can occur to code the end state of the clause, thereby licensing different types of causative clause structures.

Irish, or *Gaeilge* as it is known in the Irish language itself, is, together with Scottish Gaelic and Manx, a member of the Q-Celtic grouping of Insular Celtic. The position of the Irish language within the Celtic family of languages is indicated in figure 1. Irish is a VSO language and therefore, in common with the other Celtic languages, the order of elements in the structure of transitive sentences is verb-subject-object. The verb and the subject are tightly bound.

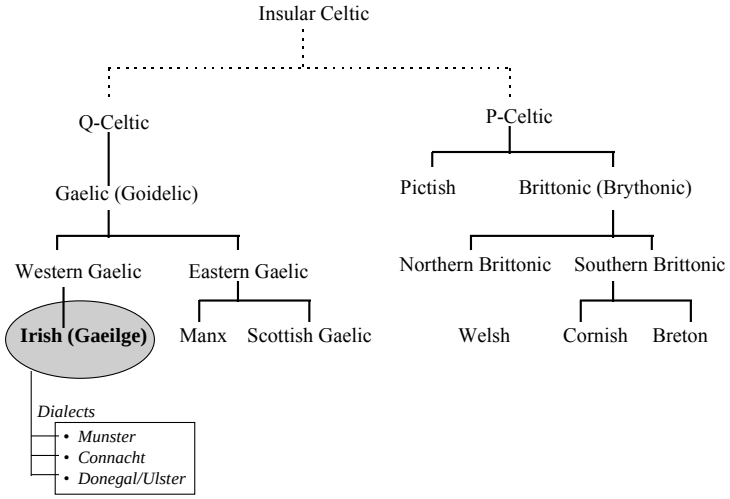


Figure 1: The Relationship Between the Celtic Languages

The functional approach in this paper makes use of many of the insights of Role and Reference Grammar (RRG) and in this paper we assume the RRG model as adequate for our purposes. Broadly, in the RRG framework (Van Valin 1993, Van Valin & LaPolla 1997) the semantic representation of sentences is based on the lexical representation of the verb. RRG employs a decompositional representation based on the theory of Aktionsart of Vendler (1967) and directly builds upon Dowty (1979, 1986, 1989, and 1991). The lexical representation of a verb or other predicate is its logical structure. The semantic representation of an argument is a function of its position in the logical structure of the predicate and the RRG linking system refers to an element's logical structure position. RRG posits two generalised semantic roles, or in Van Valin's terminology, "semantic macroroles", which play a central role in the linking system. The macroroles are actor and undergoer, and they encapsulate the usually accepted clusters of thematic roles. They are the primary arguments of a transitive predication. In an intransitive predicate, the single argument can be either an actor or an undergoer, depending on the semantic properties of the predicate.

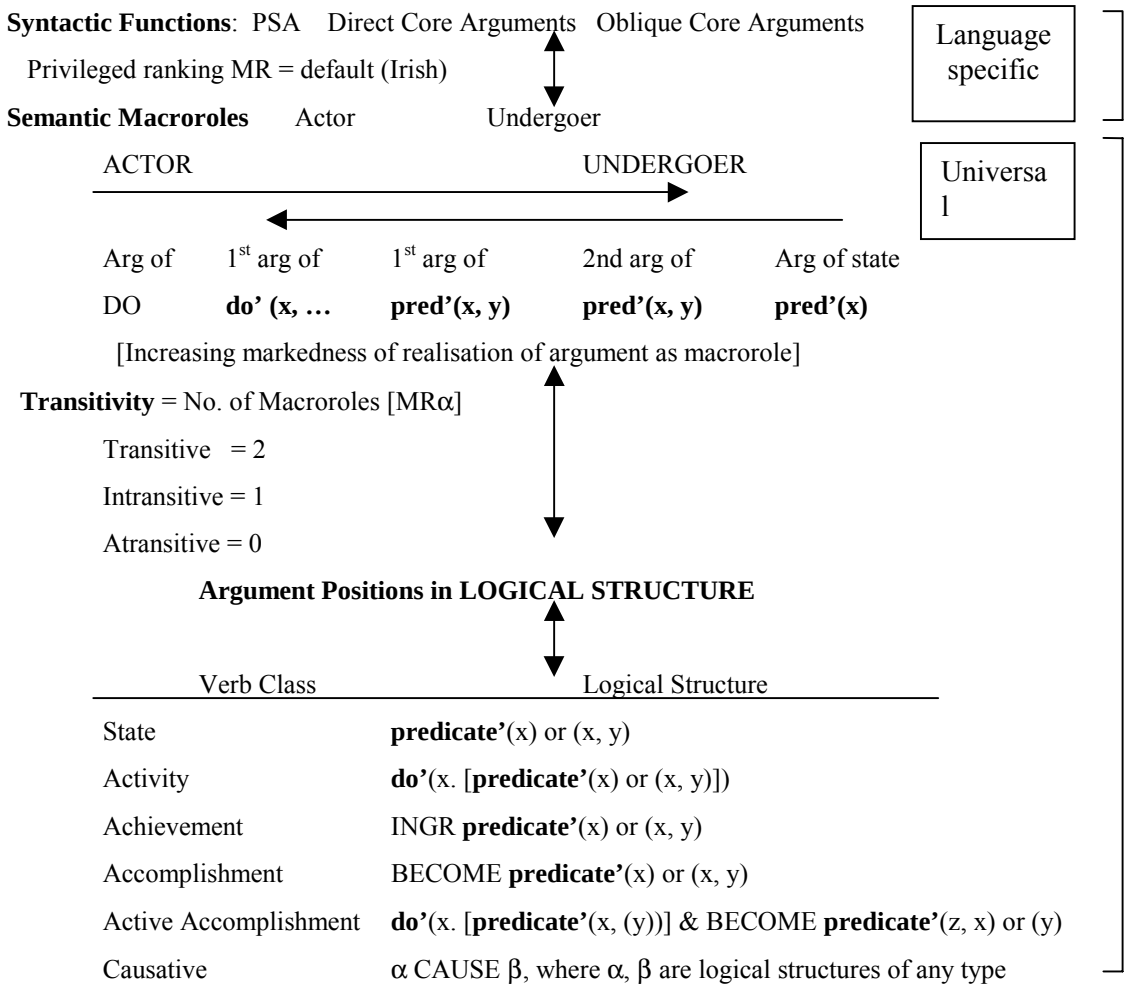


Figure 2: The System Linking the Semantic and Syntactic Representations of Irish in the RRG model (After Van Valin & LaPolla 1997).

The Actor-Undergoer Hierarchy (AUH) captures the relationship that holds between the logical structure argument positions and macroroles. In the AUH, the leftmost argument in terms of the hierarchy will be the actor and the rightmost argument will be the undergoer. Transitivity in RRG is therefore defined semantically in terms of the number of macroroles of a predicate. The algorithm that defines the linking between semantics and syntax has two phases. The first phase consists of the determination of semantic macroroles based on the logical structure of the verb (or other predicate) in the clause. The second phase is concerned with the mapping of the macroroles and other arguments into the syntactic functions.

We will demonstrate in this paper that there are a number of significant generalisations in the causative constructions that would otherwise be missed, or difficult to find, without the logical structure formalism. In particular, we deploy a decompositional representation influenced by RRG to represent the underlying situation types, states of affairs, and events to

bring out various uses of the verb *cuir* 'put' and in so doing we uncover significant evidence to support our contention that motion is a factor in causation along with the eventive primitives of CAUSE, BECOME, INGR and BE. This class of periphrastic causation that makes use of lexically causative verbs is concerned with caused motion within the event frame and is articulated over a prototypicality structure.

These ideas on prototypicality structures with a prototypical central member or base have been influential on linguistic research (Taylor 1995). In particular, Shibatani (1985: 821ff) notes that:

> "*Increasing awareness in recent years that linguistic structures are not isolated, but rather tend to show partial resemblances among themselves, has prompted certain linguists to adopt a non-discrete view of grammar. Research progress in the framework of prototype theory is one such manifestatio*n. *... This view of grammar considers that various constructions exist along a continuum; certain ones are prototypical, others are similar to the prototype to a limited degree, and still others share no similarities with the prototype. … such an approach not only is essential in understanding the relationships among various constructions within a single language, but also is capable of providing a useful framework for cross-linguistic research"*.

The sense of prototypicality as a structural notion will be used within this paper. [For prototype approaches to grammar, cf. Lakoff 1977, Hopper & Thompson 1980, Coleman & Kay 1981 and Langacker & Munro 1975]

2.0 Periphrastic Causation

By periphrastic causative constructions we mean constructions that involve the use of additional words in the clause to encode the causation, such constructions not being lexical or morphological causatives. We claim that the periphrastic causatives of modern Irish are built on a schema of caused motion (1) of an entity and encompass the full taxonomy of NP types. We will provide evidence that the periphrastic causative construction of modern Irish is sensitive to interpretation as a prototypicality structure and that the schemata in (1) represent the extensions over this prototype structure from the base.

Underlying Schemata of Caused Movement

(1) a: X CAUSES Y to MOVE to LOCATION [= Base]
b: X CAUSES STATE to MOVE to Y coded as LOCATION
c: X CAUSES Y to MOVE to STATE coded as a LOCATION
d: X CAUSES Y to MOVE to STATE

The schema allows for a wide range of causation constructions, from the caused motion of a concrete entity to an actual location, across to the caused motion of an individual into an abstract state. Both a common or proper NP may code a location. A state may be coded by an abstract mass or an abstract count NP, irrespective of whether the state is coded as a location. In the periphrastic causative construction, the nominals can therefore represent a) actual people, b) things, and c) locations, through to d) abstract states coded as locations.

The structure of this paper is as follows. After a general discussion on elements of the syntax of the construction with the lexically causative verb *cuir* 'put', and the role that particular prepositions play in licensing different situation types, we proceed with the analysis detail. We divide the analysis into four subsections, each characterising a particular schema (1). Within this the clauses are expressed in their underlying logical structure format because an event perspective is necessary to understand the interaction between the semantics and the syntax. After the analyses we summarise the evidence found.

2.1 The Template of the *Cuir* Verb and General Characteristics

The syntax of clauses that contain the verb *cuir* 'put', a verb whose lexical semantics is that of causative placement, within a periphrastic causative has a number of distinct typological attributes. In this section we briefly discuss its general characteristics. In the analysis following we discuss some representative examples that are indicative of the range of constructions involving caused motion found with this causative. The verb *cuir* 'put' is used productively in many instances of causative achievement. The underlying schemata for the argument structure in the syntax and the corresponding logical structure is:

(2) a: [*Chuir* NP_x NP_y [PP_{ar} NP_z]] Argument Structure
b: [**do'**(x) CAUSE INGR [**be'**(y, [$\mathbf{ar_{on}}$**'**(z)])]] Logical Structure
where *x: Actor participant*
y: Theme (Undergoer if z is non human)
z: Undergoer participant if the participant is animate and human, otherwise theme.

From analysing examples of these constructions that use the verb *cuir*, it is possible to discern a tendency for the constructions to deliver an inchoative interpretation of INGR, that is, an achievement, when the preposition deployed is *ar* 'on'. This also holds when the corresponding prepositional pronoun is used as a conflation of the preposition *ar* 'on' with an appropriate pronoun as [*ar*+PN]. When the construction uses a different preposition (or prepositional pronoun), for instance *i* 'in' or *leis* 'with', then the aspectual interpretation tends to be BECOME, that is, an accomplishment. The use of the construction with the

preposition *go* 'to' also tends to deliver an accomplishment. This particular preposition, when used, indicates the path of motion in the caused motion construction.

(3) [*Chuir* NP_x NP_y [$PP_{i/leis/go}$ NP_z]] Argument Structure
(4) [**do'**(x) CAUSE BECOME [**be'**(y, [$\mathbf{i_{in}}$**'**(z)])]] Logical Structure
(5) [**do'**(x) CAUSE BECOME [**be'**(y, [$\mathbf{leis_{with}}$**'**(z)])]] Logical Structure
(6) [**do'**(x) CAUSE BECOME [**be'**(y, [$\mathbf{go_{to}}$**'**(z)])]] Logical Structure

In all instances, the verb is a three-place predicate. As a template for a causative construction one finds that it is commonly used, the participants being elaborated as appropriate. In the next section we examine examples of this construction with particular reference to the types of motion and nominals involved. We will demonstrate, for example, how the undergoer is caused to move to a location, caused to move to a state, how the undergoer is the location into which some state is placed, and how the undergoer is placed into a state that is represented as a location.

2.2 Caused Movement of an Undergoer to a Location

In this section we examine clauses that demonstrate caused movement to a location. Example (7) demonstrates caused motion by an actor in which a proper NP moves from a present, but unspecified, location to another actual location represented by a proper NP. The change involved as a consequence of the caused motion is an actual change of location, not of state.

In this example (7) we have three participants in the clause. The first participant, *sé* 'he' is an animate human actor coding for agency. The next participant *Micheál Ó Cléirigh*, a proper NP, is also animate and human and is the undergoer of the verbal action. The path of motion of the action is coded by the preposition *go* 'to' and the third participant is *Éireann* 'Ireland', the goal.

(7) *Chuir sé Micheál Ó Cléirigh anall go hÉirinn.*
Put:V-PAST he:PN Micheál Ó Cléirigh:N across:ADV to:PP Ireland:N
He sent Micheál Ó Cléirigh across to Ireland.
[**do'**(sé,0) CAUSE BECOME [**be'**(Micheál Ó Cléirigh, [**go'**(Éireann)])]

In this particular causative accomplishment construction we need to assume arrival at the goal even though this is not explicitly coded. While the actor was dispatched, we have no indication of arrival. We have therefore no specific confirmation of the end condition of arrival in the new location. We can note that the preposition *go* 'to' is used with an accomplishment.

(8) *Chuir sí cóiriughadh úr-nuaidh ar an dreisiúr.*
Chuir sí cóiriughadh úr-nuaidh
Put:V-PAST she:PN ornament:N fresh:ADJ+new:ADJ
ar an dreisiúr.
on:PP the:DET dresser:N
She put a new ornament on the dresser.
[**do'**(sí) CAUSE INGR [**be'**[**úr-nuaidh'**(cóiriughadh), [**ar'**(an dreisiúr)]]]]

In example (8), the undergoer is caused to move to an actual location. The example in this clause contains three participants in its logical structure and three arguments in the syntax. The first participant is *sí* 'she', a concrete count NP and an animate human actor that is the sentence subject. The second participant is *cóiriughadh* 'ornament', a concrete count NP and an inanimate non-human entity. This participant is the undergoer and direct object. The third participant, the indirect object, is the goal at which location the undergoer is placed. In this example, we have a commitment to the end condition that results upon termination of the action. The situation type is inchoative in nature as the undergoer entity, *cóiriughadh* 'ornament', is either on *an dreisiúr* 'the dresser', a concrete count NP, or it is not. This construction is a causative achievement and we can note the preposition used is *ar* 'on'. In this example we have caused motion where the actor causes the undergoer, a concrete count NP, to move to an actual location elaborated by a concrete count NP.

2.3 Caused Movement of a State onto an Undergoer Coded as a Location

In this section we examine clauses that exhibit caused movement of a state onto an undergoer that is coded as a location. The example in (9) is typical of a caused motion construction in which an actor causes a state denoted by an abstract mass NP to move to the undergoer coded as a location.

(9) *Chuir a chuid cainnte an-iongantas go deo orm.*
Chuir a chuid cainnte
Put-V-PAST his:PN_POSS part:QTY talk:N
an-iongantas go deo orm.
much:EMP+wonder:N to:PP ever:ADV on:PP+me:PN
Lit:'His pieces of talk put much amazement on me for ever'.
His talking caused me endless amazement.
[**do'**(**a**$_1$'(cuid cainnte$_2$)) CAUSE INGR [**be'**(an-iongantas$_3$, [**ar'**(mé$_4$)])]]

The first participant is a concrete count NP, *a chuid cainnte* 'his pieces of talk', and is the instigator of the action. As such, it has the actor role in the logical structure and is the clause subject. The NP denotes the fragments of talk of an unspecified individual, an animate human. The undergoer of the action of the verb is encapsulated within the prepositional pronoun *orm* 'on+me' as a concrete count NP. This animate human participant is the clause

object. The third participant in the logical structure denotes the abstract state *an-iongantas* 'much wonder', an abstract mass NP, which will move onto the undergoer participant. This is the clause indirect object.

We show in example (10) how an abstract state, represented by the abstract mass NP, is caused to move by the actor onto the undergoer, a proper NP. The undergoer is schematically expressed as a location.

(10) *Chuir rinnce na gréine i mbrollach na mara míne draoidheacht éigin ar Mhaghnus.*
Chuir rinnce na gréine
Put:V-PAST dancing:VN the:DET-pl sun:N
i mbrollach na mara míne
in:PP bosom:N the:DET-pl sea:N smooth:ADJ
draoidheacht éigin ar Mhaghnus.
magic:N some:QTY on Maghnus:N
LIT:'The dancing of the sun in the bosom of the smooth sea put some magic on Maghnus'.
The dancing of the sun on top of the calm sea put some spell on Maghnus.
[**do'**(**i**'(**brollach na mara míne'**(**ag'**(**rince'**(na gréine)))))
CAUSE INGR [**be'**(draoidheacht éigin, [**ar'**(Maghnus)])]]

This example has a construction using the verb cuir '*put'*, a three-place predicate. In the syntax we can see three arguments. The first argument is the effector/instigator of the action and is therefore the actor. The first argument, *rinnce na gréine i mbrollach na mara míne* 'the dancing of the sun in the bosom of the smooth sea', is complex and consists of a single argument verbal noun coding a progressing activity and its internal subject argument along with a location, denoted by the preposition *i* 'in', where the progressing activity occurs.
The second argument, the undergoer participant of the logical structure of the clause is *Maghnus*, a proper NP and an animate human. In the syntax, this participant elaborates the argument within the prepositional phrase fronted by *ar* 'on' as its object. The third argument is *draoidheacht éigin* 'some magic', an abstract mass NP representing an inanimate non-human entity that is instantaneously caused to move onto *Maghnus* as a consequence of the action of the verb. This argument represents the end state. The clause codes for a causative achievement situation type that has an unbounded progressing activity as its instigator.

2.4 Caused Movement of an Undergoer to a State Coded as a Location

In this section we examine clauses that code caused movement to a state coded as a location. The example (11) following codes for caused motion by an actor whereby the undergoer, a proper NP, moves from an existing condition to a state represented by an abstract count NP and schematically expressed as a location.

(11) *Chuir sin Donnchadh ó obair*.
Put:V-PAST that:DET Donnchadh:N from:PP work:N
That put Donnchadh out of work.
[**do'**(sin) CAUSE BECOME (**be'**(Donnchadh, [**ó**'(obair)]))]

This clause is a causative accomplishment. The clause contains the verb *cuir* 'put', a three-place predicate. The logical structure of the clause has three participants and the clause has three arguments. The first participant in logical structure is *sin* 'that', a non-human entity of unspecified reference and effector of the caused action. This is the actor and clause subject. The second participant, the undergoer and direct object of the clause is *Donnchadh*, a proper NP and an animate human. The third participant is *obair* 'work' is an abstract count NP and appears in the syntax as the object argument of the preposition *ó* 'from'. In this construction, we are committed to the end state as a consequence of the specific preposition used.

The example in (12) is of a caused motion whereby the actor, a concrete count NP, causes the undergoer, a concrete count NP, to move to a state, an abstract mass NP, that is schematically expressed as a location.

(12) *Chuir sin ag smaointeadh é ar an tamall a bhí caithte i Meiriceá aige*.
Chuir sin ag smaointeadh é ar an tamall
Put:V-PAST that:DET at:PP thinking:VN him:PN on:PP the:DET time:N
a bhí caithte i Meiriceá aige.
that:REL be:SUBV-PAST spent:VA in:PP America:N at:PP+him:PP
LIT:'That put him thinking on the time that was spent by him in America'.
That set him thinking about the time that was spent by him in America.
[**do'**(sin_1) CAUSE BECOME [**be'**(**ag'**(smaointeadh'($é_2$, [**ar'**(an $tamall_3$)]))]
REL [**be'**(**caith'**(pro_3, [**i**'[Meirceá$_4$, [**ag'**($sé_2$)]]))]]

This sentence is complex in that it contains a number of clauses including a verbal noun with an embedded relative clause that, in turn, contains a personal passive construction. The primary clause has the verb *cuir* 'put', a three-place predicate requiring three participants in its logical structure and three arguments in the syntax, a subject, object and indirect object. The primary clause is a caused accomplishment that contains an embedded progressing activity and a passive voice construction.

The actor participant and sentence subject *sin* 'that', is a concrete count NP and an non-human entity of unspecified reference. The animate and human second participant, the undergoer and direct object of the syntax, is *é* 'him', a concrete count NP. The third participant of the logical structure and indirect object is *smaointeadh* 'thinking', an abstract mass NP. This participant is a verbal noun, the object of the preposition *ag* 'at', and which signifies entry into a state of progressing activity. The subject of the verbal noun is the

second participant *é* 'him'. Because this participant is elaborated by a pronoun with accusative marking it is not left positioned in its expected canonical template position but occurs in clause final position before the oblique phrases. This second participant is the object of the verb *cuir* 'put'. This clause final position is a common feature of object pronouns within Irish (Tallerman 1998: 29*ff* & 616*ff*). The object of the verbal noun is the NP *tamall* 'time'. This NP is contained within a prepositional phrase as its object *ar an tamall* 'on the time'. The verbal noun therefore has its own argument structure with a subject and object. It also contains the embedded relative clause, a personal passive construction. This passive construction uses a substantive verb and the verbal action is represented in the syntax by the verbal adjective *caithe* 'spent'. The demoted actor of this clause is downstream in oblique position and contained in the prepositional pronoun as a conflation of preposition and pronoun *aige* 'at+him'. The object of the personal passive construction is *an tamall* 'the time'. This appears in left shifted position as the object of the preposition *ar* 'on', as discussed above.

2.5 Caused Movement of an Undergoer to a State

In this section we examine clauses that code caused movement to a state. The example in (13) contains three clauses that denote a chain of events of which two form a causal chain. The first clause is a causative accomplishment. The second clause is an activity and the third clause is caused achievement. The third clause contains an instance of a noun used as a verbal predicate. The event of the third clause is a direct causal consequence of the action denoted within the second clause such that:

(13) *Chuir sé an chéasla trasna ar thoiseach an churraigh,*
tharraing air a phíopa agus dhearg é.
Chuir sé an chéasla trasna
Put:V-PAST he:PN the:DET paddle:N across:ADV
ar thoiseach an
on:PP breadth:N (of) the:DET
churraigh, tharraing air a phíopa agus
currach:N pull:V-PAST on:PP his:PN_POSS pipe:N and:CONJ
dhearg é.
redden:V-PAST it:PN
He put the oar across the width of the currach, pulled on his pipe and reddened it.
[**do'**(sé$_1$) CAUSE
BECOME [**be'**(an chéasla$_2$, [**trasna'**[**ar'**(thoiseach an churraigh$_3$)]])] &
[**do'**(pro$_1$, [**tharraing'**(pro$_1$, [**ar'**(**a**'(píopa$_4$))])])] &
[**do'**(pro$_1$, [**dearg'**(pro$_1$, é$_4$)]) CAUSE INGR [**be'**(**dearg'**(é$_4$)]]

(14) [Clause 1]$_{\textbf{Causative Accomplishment}}$ & [[Clause 2]$_{\textbf{Activity}}$ → [Clause 3]$_{\textbf{Caused Achievement}}$]

The verb in the first clause is *cuir* 'put', a three-place predicate. The first participant is *sé* 'he' an animate human entity and the volitional instigator of the action. This participant is the clause subject. The second participant is *an chéasla* 'the paddle', a concrete count NP and a non-human and inanimate undergoer that is the direct object. The third participant is *toiseach an churraigh* 'the width of the currach'[3], a concrete count NP. This participant is fronted by the preposition *ar* 'on'. Because the verb is transactional the third participant is always the object of a preposition, as we see with *ar toiseach an churraigh* 'on the width of the currach'. In this clause we have an example of caused motion where the actor causes the undergoer to move to a location.

The second clause has the verb *tharraing* 'pull' and one argument in the syntax. However, while the clause is syntactically intransitive, the logical structure of the verb in the clause reveals that it is semantically transitive having two participants. The participant that is not expressed in the syntax is the actor. This is represented by pro_1 in the logical structure and co-refers to the participant that elaborates the subject argument in the first clause. The single argument expressed in the syntax is fronted the preposition *air* 'on' and by a possessive pronoun *a* 'his' which co-refers to the subject argument in the first clause *sé* 'he'. The syntactic single argument of this clause is *píopa* 'pipe', a concrete count NP and a non-human inanimate participant, and the undergoer of the logical structure.

The third clause contain a single syntactic argument, the pronoun *é* 'it' which co-refers to the syntactic single argument of the previous clause *píopa* 'pipe'. The pronoun has accusative marking suggesting that the verb, as used in the clause, requires two participants within the logical structure. The missing argument in the syntax is the actor participant in the logical structure. This is expressed in the logical structure by pro_1 and co-refers to the actor participant of the logical structure of the first clause. This clause is an example of causation whereby the undergoer is moved to a state denoted by an abstract mass NP. In this example this process is lexicalised as a verb. The third clause is therefore also interesting for this reason. The predicate *dearg* is normally considered to be in the first instance a noun. The predicate *dearg* can also be used as an adjective to denote an attribute of some nominal entity. When *dearg* is used as an adjective it must appear immediately post adjacent to the right of the NP that it is associated with. In the third clause of our example *dearg* is used as a verbal predicate. To be used in this manner the verb requires a syntactic argument structure

[3] A *currach* is a type of boat used along the Atlantic coast of Ireland.

that is motivated by an underlying lexical semantics, which we represent with the logical structure representation below.

(15) [*dearg*:N] 'red'.

(16) [x:N *dearg*:ADJ] '(a) red x'.

(17) [*dearg*:V x:NP y:NP] Syntactic Argument Structure

(18) [**do'**(x, [**dearg'**(x, y) CAUSE INGR [**be'**(dearg'(y))] Logical Structure

where: x: Actor participant

y: Undergoer participant

The use of *dearg* as a verbal predicate denotes the process whereby a specified undergoer entity receives a certain state, and that state being described by the N or ADJ form. An entity either is, or is not, *dearg*. The caused change of state denoted by the verbal predicate is therefore inchoative and lexically, the situation type is an achievement.

In (19) we have caused motion where an unspecified actor, that we know to be human and animate, causes the undergoer, a concrete count NP, to move into a state denoted by an abstract mass NP.

(19) *Chualathas anois ceol na bpíob ag teacht ó dhá cheann na sráide agus chuir sin fuillsceadh faoi a raibh ag éisteacht*.
Chualathas anois ceol na bpíob
Hear:V-Impers-Pass-PAST now:ADV music:N the:DET pipes:N
ag teacht ó dhá cheann na sráide
at:PP coming:VN from:PP two:NUM head:N the:DET street:N
agus chuir sin fuillsceadh
and:CONJ put:V-PAST that:DET passion:N
faoi a raibh ag éisteacht.
under:ADV that:REL be:SUBV-PAST at:PP listening:VN
Lit:'(One) now heard the music of the pipes coming from both ends of the street and it put passion into all that were listening'
People now heard the music of the pipes coming from both ends of the street and it put passion into all that were listening.
[**anois'**[**do'**(x_1, [**chuala'**(x_1,
[**ó**'(**dhá cheann na sraide$_3$'**([**ag'**(**teach'**(ceol na bpíob$_2$))]))])])]
& [**do'**(sin_2) CAUSE
BECOME (**be'**(**faoi'**(**be'**(**ag'**(**eisteacht'**(x_1), fuillsceadh$_4$)))))]]

This complex sentence has two clauses. The first clause is an impersonal passive construction with an impersonal actor not expressed in the syntax as an argument, and a direct object *ceol na bpíob* 'music of the pipes'. The NP *ceol na bpíob* 'music of the pipes' is also the subject of the oblique argument fronted by the preposition and verbal noun *ag teacht* 'at+coming'. The verbal noun denotes a one-place predicate. The logical structure of the clause indicates

that the role of the actor of the impersonal is not elaborated but is visible to the syntax. Typically, this impersonal actor is human and animate but remains specific and indefinite.

The second clause after the conjunction *agus* 'and' codes for a causative accomplishment and uses the three place transactional predicate *cuir* 'put'. The first participant in the logical structure of this clause is *sin* 'that'. In this instance, *sin* 'that' refers to the participant in the logical structure of the preceding clause *ceol na bpíop*. The participant in the undergoer role is complex. It is denoted by a relative clause that contains a substantive verb and a verbal noun. The substantive verb and verbal noun combination code for a progressing activity. The actor of this inner clause with the unbounded activity is not specified in the syntax, as such, but is co-referential with the impersonal actor of the first clause. The third participant of the second clause denotes the abstract state *fuillsceadh* 'passion', an abstract mass NP and the end result of the caused action. That is, the state into which the undergoer moves.

3.0 Summary of Periphrastic Causatives

In this paper we have provided an analysis of patterns of a class of periphrastic constructions that employ lexical causatives verbs in co-occurrence relations with a bounded set of prepositional phrases, within which the underlying schema (20) is that of caused movement. In particular, we concentrated on the verb *cuir* 'put', a verb whose lexical semantics is that of causative placement. The resulting periphrastic causative constructions exhibited a polysemy on the verb beyond that of its lexical definition of causative placement. The schema allows for a wide range of causation constructions, from the caused movement of a concrete entity to an actual location, across to the caused movement of an individual into an abstract state.

Underlying Schema of Caused Movement

(20) a: X CAUSES Y to MOVE to LOCATION [= **Base**]
b: X CAUSES STATE to MOVE to Y coded as LOCATION
c: X CAUSES Y to MOVE to STATE coded as a LOCATION
d: X CAUSES Y to MOVE to STATE

Our research findings are summarised in the table (21). The participants denoted as X and Y in the schemata range over the NP types indicated within the table. Within this, a common or proper NP may code the location. The state may be coded by an abstract mass or abstract count NP, irrespective of whether the state is coded as a location or not. The table indicates in summary form whether a state may be coded, a location may be coded and whether the state may be coded as a location, along with type of NP deployed.

(21)

Actor	Undergoer	State Coded	Location Coded
CCNP	P-NP	no	P-NP
CCNP	CNP	no	CCNP
CCNP	CCNP	AMNP	no
CCNP	CCNP	ACNP	no
CCNP	ACNP	ACNP	yes
CCNP	P-NP	ACNP	yes
CCNP	CCNP	ACNP	yes
CCNP	ACNP	no	CCNP
CCNP	P-NP	AMNP	yes
ACNP	CCNP	AMNP	yes
AMNP	CCNP	AMNP	yes

where: CCNP Concrete count NP ACNP Abstract count NP
CMNP Concrete mass NP AMNP Abstract mass NP
P-NP Proper NP

We have provided evidence that this class of periphrastic causative constructions of modern Irish, that employ lexically causative verbs, is sensitive to interpretation as a prototypicality structure and that the schemata represent the extensions over this from the base of (20a). This evidence supports our argument of the caused motion schema.

4.0 References

Coleman, Linda and Kay, Paul. (1981). Prototype semantics. *Language* 57. 26-44.

Croft, W. (1990a). Possible Verbs and the Structure of Events. In Tsohatzidis, S.L. (ed.). *Meanings and Prototypes: Studies in linguistic categorisation*,

Hopper, Paul and Thompson, Sandra. (1980). Transitivity in Grammar. *Language* 56.251-99.

Lakoff, G. (1977). Linguistic gestalts. *CLS* 13. 236-87.

Langacker, Ronald W. and Munro, Pamela. (1975). Passives and their meaning. *Language* 51: 789-830.

Shibatani, Masayoshi. (1985). Passives and related constructions: a prototype analysis. In *Language*, volume 61 number 4, pp. 821-848.

Siewierska, Anna (ed.). (1998). *Constituent Order in the Languages of Europe.* Empirical Approaches to Language Typology. Eurotype 21-1. Mouton de Gruyter, Berlin.

Song, Jae Jong. (1996). *Causatives and Causation: A Universal-Typological Perspective*. Longman, London and New York.

Stenson, Nancy. (1981). *Studies in Irish Syntax*. Narr, Tubingen.

Tallerman, Maggie. (1998a). Celtic Word Order. In Siewierska, Anna (ed.). *Constituent word order in the Languages of Europe*, pp. 21-45. Empirical Approaches to Language Typology. Eurotype 21-1. Mouton de Gruyter, Berlin.

Tallerman, Maggie. (1998b). Word Order in Celtic. In Siewierska, Anna (ed.). *Constituent word order in the Languages of Europe*, pp. 599-647. Empirical Approaches to Language Typology. Eurotype 21-1. Mouton de Gruyter, Berlin.

Talmy, Leonard. (1975). Semantics and Syntax of Motion. In Kimball, John (ed.). *Syntax and Semantics No. 4*, pp. 181-298. Academic Press, New York.

Talmy, Leonard. (1976). Semantic Causative Types. In Shibatani, Masayoshi (ed.). *Syntax and Semantics No. 6: The Grammar of Causative Constructions,* pp. 43-116. Academic Press, New York.

Talmy, Leonard. (1978). Figure and Ground in Complex Sentences. In Greenberg, Joseph H. (ed.). *Universals of Human Language, Volume iv: Syntax*, pp. 627-649. Stanford University Press, Stanford.

Talmy, Leonard. (1985). Lexicalisation Patterns: Semantic Structure in Lexical Forms. In Shopen, T.

(ed.). *Language Typology & Syntactic Description 3*: *Grammatical Categories and the Lexicon*, pp. 57-149. Cambridge University Press, Cambridge MA.

Talmy, Leonard. (1988). Force Dynamics in language and cognition. *Cognitive Science* 12:49-100.

Talmy, Leonard. (1996a). Windowing of attention in language. In Shibatani, Masayoshi and Sandra A. Thompson (eds.). *Grammatical Constructions*: *their form and meaning*, pp. 235-287. Clarendon Press, Oxford.

Talmy, Leonard. (1996b). Fictive motion in Language and "Ception": The Emanation Type. In Bloom, P. et al. (eds.), *Language and Space*, pp. 211-275. MIT Press, Cambridge MA.

Talmy, Leonard. (2000). *Towards a Conceptual Semantics*, Volume 1 and 2. MIT Press, Cambridge MA.

Taylor, John R. (1995). *Linguistic Categorization: Prototypes in Linguistic Theory*, 2nd edition. Oxford University Press, Oxford.

Tsohatzidis, S. L. (ed.). (1990). *Meanings and Prototypes: Studies in linguistic categorisation.* Routledge, New York and London.

Van Valin, Robert D. and LaPolla, Randy J. (1997). *Syntax: structure, meaning and function.* Cambridge textbooks in linguistics. Cambridge University Press, Cambridge MA.

Vendler, Z. (1967). *Linguistics in Philosophy.* Cornell University Press. Ithaca, New York.

Web Services Technology Infrastructure

Geraldine Gray & Kieran O'Connor (2002)

School of Informatics and Engineering
Institute of Technology Blanchardstown, Dublin, Ireland

Kieran.O'Connor@itb.ie

Abstract

Web Services using eXtensible Markup Language (XML) based standards are becoming the new archetype for enabling business to business collaborations. This paper describes the conceptual architecture and semantics of constructing and consuming Web Services. It describes how Web Services fit into the enterprise application environment. It discusses Web Services security. Finally, it outlines the flaws of Web Services in their current state.

Introduction to Web Services

> '*A Web Service is an application that accepts requests from other systems across the Internet or an Intranet, mediated by lightweight, vendor-neutral communication technologies*' [Kao, 2001, section II: Introduction]

Web Services expose business functionality over a network. It is not the exposed functionality that makes Web Services revolutionary; it is the lightweight, vendor neutral communication technologies. XML based standards to which vendors and developers will comply such as SOAP1.1 (and soon SOAP1.2) makes the process of inter-system communication easier. Particularly from a business-to-business perspective, Web Services as they stand today offer a huge technology leap in web enabling back end business logic.

Usually, for two businesses to share data, complex systems and communication agreements would potentially have to be developed because of the differing proprietary applications within each company. With Web Services the interaction layer is abstracted away so that developers can concentrate in providing back end functionality with the knowledge that such functionality can potentially be web enabled because Web Services will define the communication mechanisms in an interoperable way.

The most important industry defined standards that allow for interoperable communication are as follows:

- SOAP and WSDL: World Wide Web Consortium (W3C).
- UDDI: UDDI.org.

These standards define how to publish, describe and invoke a Web Service irrespective of the underlying operating system. SOAP (see page 5) defines how messages are transmitted. WSDL documents define how services are described. UDDI defines how to maintain Web Services and company information within an XML based registry.

Market Place

The market place today is influenced by many diverse companies. The contenders include Sun, IBM, Microsoft, and IONA among others. All these companies offer either Web Services toolkits or are involved in developing application servers that work with Web Services. These toolkits and Web Services compliant application servers are based upon the standards such as SOAP and WSDL. Each has its own implementation but because of the interoperable nature of Web Services most development should be portable between different proprietary software.

Sun has its SunONE Web Services architecture based on XML and Java, including its own Integrated Development Environment (IDE) and incorporating the iPlanet server family. This provides Web Services functionality to the J2EE environment. Microsoft has developed its .NET framework which provides support for web services using Microsoft specific technologies such as Visual Basic, C#, and the Microsoft server family. Other application servers have been developed with Web Services in mind, including Macromedia's JRun4.0, BEA's WebLogic, and Systinet WASP Server for Java to name but a few. There could be as many as twenty five or more SOAP APIs on the market today judging by the interoperability tests carried out by Apache AXIS. Open source projects supporting Web Services include: Apache's SOAP API AXIS and the JBoss application server.

Companies that have already implemented Web Services solutions include Amazon and Google. Amazon has produced a service that allows programmatic access to their catalogue, search engine, shopping cart and merchandising tools. Google has produced a service that allows programmatic access to their World Wide Web search engine. These services allow their functionality to be incorporated into any Web Service application.

Benefits & Shortcomings

There are a number of advantages and disadvantages to using Web Services over other similar architectures such as CORBA. Web Services claim to be platform independent, loosely coupled and can navigate firewalls over the HTTP transport protocol. Web Services

have industry backing which will facilitate their evolution. CORBA IIOP on the other hand is not platform independent. CORBA IIOP usually requires an infrastructure which includes a CORBA ORB; this limits developers to vendors who support CORBA [Monson-Haefel, 2002, p3]. A distributed application using CORBA is tightly coupled meaning that A must know about B and vice versa for the creation of Interface Definition Language (IDL) files to map data types. CORBA IIOP has no protocol specification for firewall navigation [De Jong, 2002, p4]. Disadvantages of Web Services include performance and security. Web Services rely on XML parsers to understand a particular message. This process can be slow depending on the size of the message being parsed. HTTP can be slow and unreliable as a method for transporting SOAP messages. Security is not yet well defined for Web Services. Some standards have been released and others are in the pipeline; it could be some time however before vendor support for these standards is realised. There are multiple SOAP engine implementations which can result in non portable code. Apache run tests to check how the popular SOAP implementations respond to AXIS clients which highlight these incompatibilities. Thus, Web Service code is not truly portable despite the standardisation of SOAP.

A Conceptual Model

The classic Web Services architecture defines three roles [Chappell & Jewell, 2002, pp 14-23].

- Service Requester
- Service Provider
- Service Registry/Broker

The Service Provider is the organisation responsible for creating the Web Service. First the business logic which possibly accesses databases and/or legacy/ERP systems is created. They then expose this functionality as a Web Service by publishing the organisation and Web Service details to a global registry (UDDI or ebXML). The Service Requester is the organisation that requires the use of the functionality exposed by the Web Service in question. This organisation queries the global registry to find and use a suitable Web Service. The Service Registry/Broker is an XML based data store for information including company name, contact information, and pointers to a Web Services Description Language (WSDL) file that details how to use a particular service.

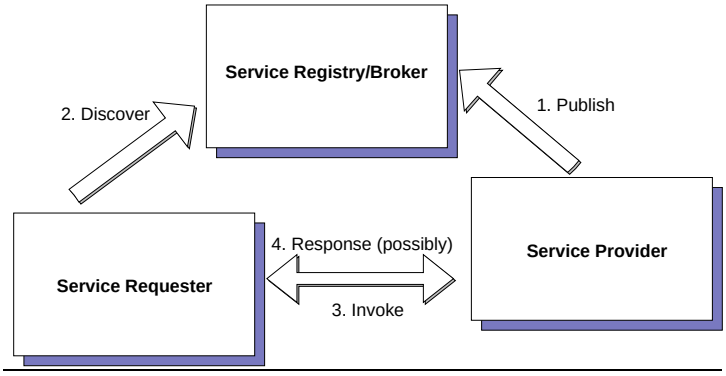


Figure 1 – Web Service Publish & Discovery Architecture.

Standards Overview

There are three major technologies that have enabled the development of Web Services and more importantly, enabled them to address the problems posed with integrating remote systems. These standards define the data transport mechanisms, how to describe what is being transported, and how to make it easy to locate Web Services. These standards are XML based, which is a universally accepted textual data format.

SOAP

The *Simple Object Access Protocol* is a lightweight distributed computing protocol that allows information to be exchanged in a decentralised environment [Hendricks et al., 2002, pp 33-61]. SOAP ensures interoperability between differing application environments by defining a messaging standard. A C++ client (for example) could create a SOAP message and send it to a remote Java Web Service over HTTP, invoking a remote procedure call (RPC) on that service. The Java Web Service would understand this request as it understands SOAP and could return a result, again encoded within a SOAP message.

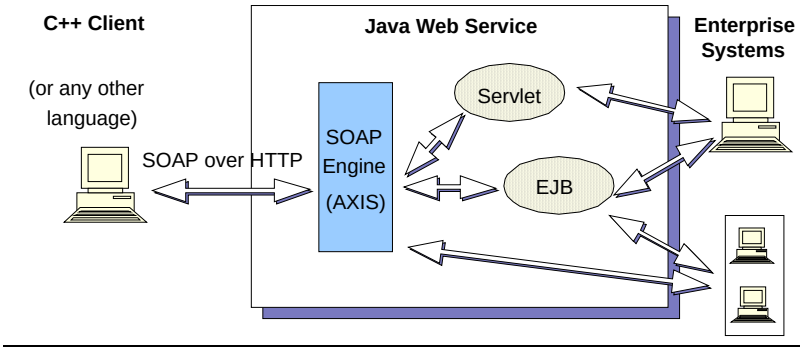


Figure 2 – Web Services Integration, through SOAP, with Enterprise Applications.

A SOAP message can have four basic sections:

- Transport Information: Transport protocol specific information. This section also contains a SOAPAction element. This element defines the intent of a Web Service call. However, the first section of the HTTP header, i.e., the POST section, also defines the intent of the service call. Hence, the SOAPAction is generally left blank. The use of this SOAPAction element is still an on-going debate.
- <soapenv:Envelope>: The SOAP envelope (mandatory) defines SOAP message boundaries, i.e., where the SOAP message begins and ends. The envelope usually contains the following format directives. The <soapenv:encodingStyle> specifies the structure of the SOAP message. The <xmlns:soapenv> specifies the structure of the envelope element of a SOAP message. <xmlns:xsd> and <xmlns:xsi> specify the XML schema instance namespace and XML schema namespace respectively. These namespaces will define tag meaning etc.
- <soapenv:Header>: SOAP header (optional) provides directives for the SOAP processor, there are no rules as to what should be included in this section. Seen as a way to add features such as basic security, transaction management, and payment services to a SOAP message [Hendricks et al., 2002, p 111].
- <soapenv:Body>: The SOAP body (mandatory) contains the actual data such as service name, parameter values and method names.

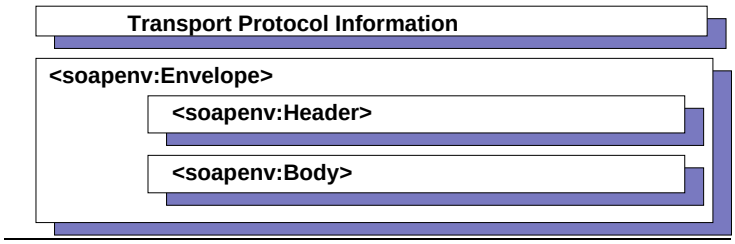


Figure 3 – SOAP Message Structure.

The following is an example of a SOAP request over HTTP. This message sends a complex type (Java Bean) called *Widget* consisting of two strings to the Web Service called *ServiceName* invoking the method *processWidget* on class *WidgetService*.

```
POST /axis/servlet/AxisServlet HTTP/1.0
Host: localhost
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Content-Length: 800

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
            <soapenv:Body>
                  <ns1:processWidget xmlns:ns1="ServiceName">
                        <arg1 href="#id0"/>
                  </ns1:processWidget>
<multiRef id="id0" SOAP-ENC:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="ns2:Widget" xmlns:ns2="urn:WidgetService">
                        <colour xsi:type="xsd:string">Blue</colour>
                        <size xsi:type="xsd:string">24</size>
                  </multiRef>
            </soapenv:Body>
</soapenv:Envelope>
```

WSDL

The *Web Services Description Language* describes a Web Service in a universally understandable way [Hendricks et al., 2002, p 148]. The WSDL file associated with a Web Services describes where the service is located, what operations are available, how to invoke those operations (parameters), and what transport protocol to use. WSDL is based on XML to ensure interoperability between platforms. An example is a Java Web Service exposing a WSDL file describing itself; a C++ (or any other language) client that is XML/WSDL aware can parse and understand this document and hence understand how to invoke the Web Service.

A WSDL file can have seven basic sections:

- <definitions>: Like the SOAP envelope, this delimits the beginning and end of a WSDL file. XML namespaces are also defined. These include namespaces to define the WSDL framework, WSDL SOAP binding, and the XML schemas.
- <types>: The data types used.
- <message>: The request and response messages exchanged.
- <operation>: Defines the actual methods to be invoked and the invocation style to use. This element contains the <input> and <output> tags which define how the request and response messages are to be encoded.
- <portType>: Collection of related operations.
- <binding>: Defines the communication protocol to be used, e.g., SOAP over HTTP.
- <service><port>: Location of Web Services, i.e., the endpoint Uniform Resource Indicator (URI).

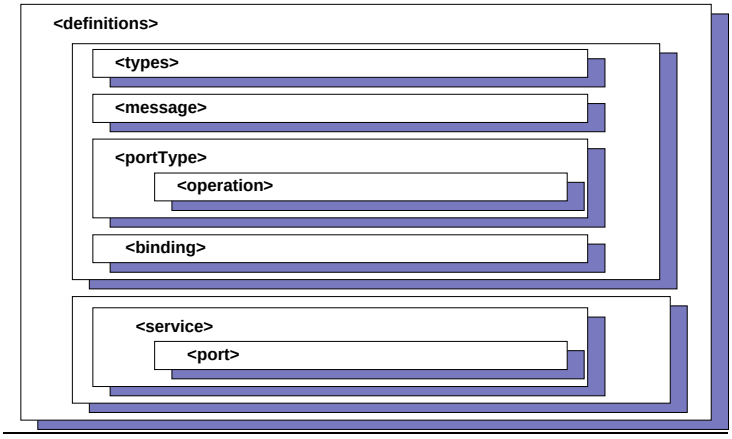


Figure 4 – WSDL Message Structure

The following is an example of a WSDL file. The request parameters consist of an integer and a string. The response is an integer. There is one operation available: *reserve*, which takes the two parameters as input. The invocation style will be *RPC* over *HTTP*. The service is accessible through the endpoint: *http://127.0.0.1:8101/compass/services/Reservation*.

```
      <?xml version="1.0" encoding="UTF-8"?>
      <wsdl:definitions
targetNamespace="http://127.0.0.1:8101/compass/services/Reservation"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:intf="http://127.0.0.1:8101/compass/services/Reservation"
      xmlns:impl="http://127.0.0.1:8101/compass/services/Reservation-
impl" xmlns:SOAP-  ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
            <wsdl:message name="reserveRequest">
                  <wsdl:part name="in0" type="xsd:int"/>
                  <wsdl:part name="in1" type="xsd:string"/>
            </wsdl:message>
            <wsdl:message name="reserveResponse">
                  <wsdl:part name="return" type="xsd:int"/>
            </wsdl:message>
            <wsdl:portType name="Reservation">
            <wsdl:operation name="reserve" parameterOrder="in0 in1">
      <wsdl:input message="intf:reserveRequest"/>
      <wsdl:output message="intf:reserveResponse"/>
            </wsdl:operation>
            </wsdl:portType>
            <wsdl:binding name="ReservationSoapBinding"
                        type="intf:Reservation">
            <wsdlsoap:binding style="rpc"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
            <wsdl:operation name="reserve">
      <wsdlsoap:operation soapAction="" style="rpc"/>
      <wsdl:input>
            <wsdlsoap:body use="encoded"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="http://127.0.0.1:8101/compass/services/Reservation"/>
      </wsdl:input>
      <wsdl:output>
            <wsdlsoap:body use="encoded"
      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="http://127.0.0.1:8101/compass/services/Reservation"/>
      </wsdl:output>
            </wsdl:operation>
```

```
        </wsdl:binding>
        <wsdl:service name="ReservationService">
        <wsdl:port name="Reservation"
                    binding="intf:ReservationSoapBinding">
<wsdlsoap:address
location="http://127.0.0.1:8101/compass/services/Reservation"/>
        </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

UDDI

The *Universal Description Discovery and Integration* specification provides a framework for publishing Web Services and for the discovery and consumption of those services by interested parties [Hendricks et al., 2002, pp 151-193]. The UDDI specification defines a global registry structure for holding XML based data. UDDI is not designed specifically with Web Services in mind; it is also a registry for general company information. Note also that while UDDI is the current leader in XML based registries, electronic business XML (ebXML) is fast gaining ground. This is a superset of the UDDI specification and includes more features such as Business Process Management. A provider can publish a Web Services to a registry by providing business information such as the company name and a technical specification of the Web Service(s) provided including the location of a defining WSDL file. A Web Service requester can then discover a Web Service based on search criteria.

The publish and discovery process can be done manually or using the Java API for XML Registries (JAXR) developed by Sun. Registry's can be accessed manually, e.g., a website, (IBM's test registry: http://www.ibm.com/services/uddi/testregistry/protect/registry.html), or using some other proprietary tool. JAXR provides programmatic interfaces through which access to XML based registries are possible. Private registries can be used and are useful for intranets and local corporate networks where Web Services are not being made globally available.

The following is an example of the information contained within a SOAP message to query a registry and the returned result. The query can be created and executed using the JAXR API. For brevity, some of the result information and the SOAP message specifics have been omitted. The query on the UDDI registry is done on business name 'Microsoft'. This returns an XML representation of all the information contained about Microsoft in the particular registry. While the query given in this instance is find_business, other queries can be given such as find_service and find_relatedBusiness.

Query:

```
<uddi:find_business generic="1.0" xmlns="urn:uddi-org:api">
                <uddi:name>Microsoft</uddi:name>
</uddi:find_business>
```

Result:

```
<businessList generic="1.0 operator="Microsoft Corporation"
truncated="false" xmlns="urn:uddi-org:api">
  <businessInfos>
    <businessInfo businessKey="0076B468-EB27-42E5-AC09-
9955CFF462A3">
            <name>Microsoft Corporation</name>
            <description xml:lang="en">
                          . . .
            </description>
            <serviceInfos>
                    <serviceInfo
                      businessKey="0076B468-EB27-42E5-AC09-99"
                      serviceKey="1FFE1F71-2AF3-45FB-B788-0A4">
                                  <name>. . .</name>
                            </serviceInfo>
                    </serviceInfos>
            </businessInfo>
        </businessInfos>
 </businessList>
```

The structure of the data held within a UDDI registry consists of five elements [Chappell & Jewell, 2002, pp 104-106]:

- <businessEntity>: The items contained within this element describe business information such as name and address.
- <businessService>: Describes the services offered by the business.
- <bindingTemplate>: Contains pointer to technical descriptions of services, their associated URL, and possibly a textual description of the service.
- <tModel>: Contains the specifics of how to interact with a Web Service including a pointer to the WSDL file.
- <publisherAssertion>: Outlines business-to-business relationships.

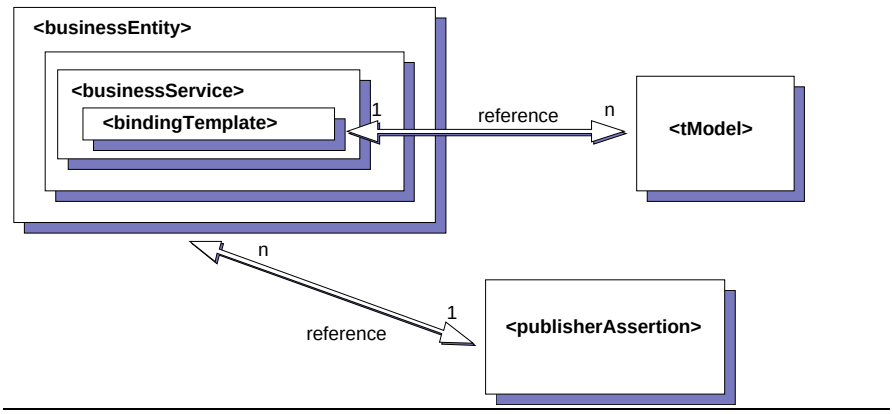


Figure 5 – UDDI Data Structure.

Clients

Currently there are two workable ways for a client wishing to use a Web Service to invoke that service.

Local stub

This involves using a tool to generate proxy stubs [Chappell & Jewell, 2002, p 166]. The tool typically reads a WSDL file and outputs the required proxy/stub files for interaction with the Web Service. This is probably the most common and convenient method used. Almost all vendors will provide such tools with their Web Services toolkits.

The use of stubs allows us to interact with a remote object, through the stub, as if they were local to our runtime. The stubs take care of details such as connecting to the remote machine and data types. Essentially the stub class would act as the Web Service and we can just call methods directly on this class. Using the same tools, skeleton files can also be created for use on the server side. Example tools include:

- proxygen: IBM
- xrpcc: Sun
- wsdl2java: Apache (AXIS)

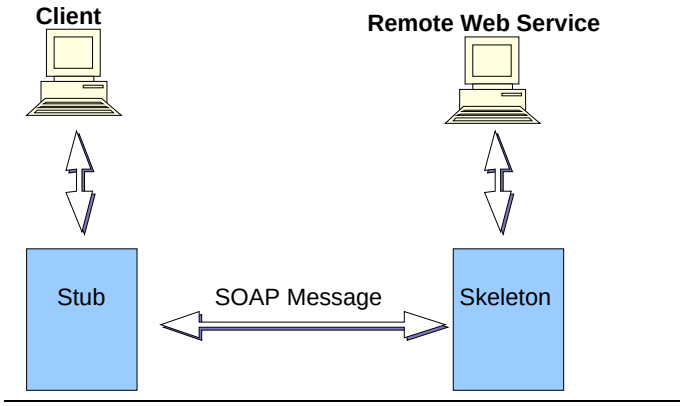


Figure 6 – Stub File Web Service Interaction.

Dynamic Invocation Interface (DII)

DII uses the *javax.xml.rpc.Call* object, which is part of the JAX-RPC, to build up parameters to be sent to a web service [Chappell & Jewell, 2002, pp 166-171]. Method name, method parameters, service endpoint, and return type are all set in the object and an invocation takes place. Generic objects are passed between the client and the servers and cast to the more specific objects on arrival. This is less efficient than the local stub method which deals with

actual objects. Clients are more difficult to code, debug, and test. However it is useful for one way RPC, for when services are discovered dynamically, or where stub creation tools may not be available.

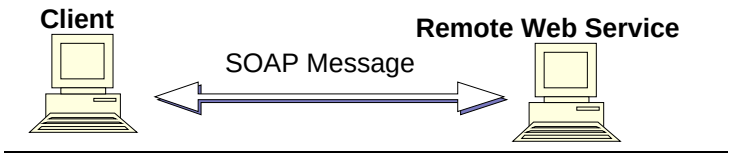


Figure 7 – Dynamic Web Service Invocation.

Security

One of the main concerns with any application made available over a network is security. There are many efforts aimed at developing security standards for various aspects of Web Services. However it should be noted that these standards are immature and clear standards for securing web standards are still a thing of the future.

Concern

Web Services expose many obvious security risks, particularly due to the fact that SOAP using HTTP on port 80 can send requests through corporate firewalls. Hence many traditional security measures are bypassed.

Solutions

Current efforts are aimed at standardising the way in which Web Services will be secured. These efforts include SOAP Digital Signature and XML Encryption. However these standards have either been released recently or are still at the specification stage meaning vendor support may not be realised for some time to come. For example, the IBM Security Toolkit provides an implementation of the XML Encryption standard, but this is just an experimental reference implementation.

Web Services security can be broken down into two distinct areas:

1. Back end web application security: can leverage existing application security frameworks such as the Java Authentication and Authorisation Service (JAAS).
2. Communication layer security: focuses on the transmission of data and hence incorporates such security mechanisms as data encryption/decryption and transport layer security protocols such as Secure Sockets Layer (SSL).

While organisations such as W3C and the Organisation for the Advancement of structured Information Standards (OASIS) work on Web Service security standards, Web Services will be secured using traditional transport layer protocols such as SHTTP and SSL. Also, although SOAP can navigate through firewalls, a certain level of intelligence can be built into firewalls to check the content of the SOAP message and do authentication/authorisation based on that.

For the big players with big budgets, packages exist that offer Web Services security 'frameworks' providing a patchwork of security features specifically tweaked for use with Web Services. For example, Quadrasis has an Enterprise Application Security Integration system called EASI Security Unifier which does just that. However, these proprietary security systems are often expensive. Some organisations will require such stringent identification, authentication, authorisation, integrity, privacy and non-repudiation checks for securing critical data, others will either carefully review what their Web Services expose, or wait for security standards definitions to advance.

Future

Web Services are the way forward for many organisations wanting to expose functionality to a wider audience over a network. Through its standards based approach it will receive widespread buy-in from vendors and developers. However, in its current immature state, it has its flaws. Many of the security standards are still at specification stage.

OASIS has formed a technical discussion group to open up the floor to the *WS-Security* standard as a trusted means for applying security to Web Services. First published in April 2002 as part of a working partnership between Microsoft, IBM, and VeriSign, the WS-Security specification defines a standard set of SOAP extensions, or message headers, which can be used to set security technologies such as encryption and digital signatures, for instance, onto Web Services applications. The first meeting of the technical committee was held during the first week of September.

Liberty Alliance is another movement towards defining web application security. As outlined on the Liberty Alliance Project website, the objectives of the project are to:

- Develop specifications that enable service providers to protect consumer privacy.

- Provide an open single sign-on specification that includes federated authentication from multiple providers operating independently.
- Enable organisations to control, maintain and enhance relationships.
- Create a network identity infrastructure that supports all current and emerging network access devices.

Bloor Research [July 2002], identifies seven outstanding issues that need to be addressed before Web Services can be considered an enterprise class distributed systems architecture. These are: security/privacy, messaging/routing, quality-of-service/reliability, transaction processing, management, performance, interoperability. While these are stumbling blocks for the advancement of Web Services, generally there exist vendor products and solutions to address these problems [Bloor Research, 2002, p11].

Also on the horizon is the concept of 'smart' Web Services. These are services that can maintain a shared context, have multinet capabilities and have quality of service metrics. A shared context involves understanding under what conditions it was invoked. Multinet capabilities relate to a Web Services ability to receive requests from multiple device types, e.g., PDAs, mobile phones, and so on. Quality of service metrics relates to the reliability of communicated information; this encroaches upon the territory of security and reliable messaging [Hendricks et al., 2002, p 471].

Research

As Web Services are still in such an early stage of development, producing real-world applications of this technology is a current challenge within the IT community. Web Services opens up many areas of possible research including streamlined integration and web enabling legacy applications. Industries that have traditionally stayed away from technological advancement may be convinced to adopt Web Services as a means of simplifying many paper driven processes and legacy systems integration. For example, sections of the financial services sector will look to reduce cost by automating as many of their business processes as possible. Web Services are the ideal technology to allow businesses to provide services to outside parties thus delegating work elsewhere resulting in reduced operating costs. Web Services are also the ideal technology for allowing disparate systems integration.

Conclusion

With Web Services we are moving to service oriented development. Among the main principals of Web Service oriented development [Bloomberg, 2002, all] are:

- **Dynamic services replace static components**. Through WSDL the location of particular services can change dynamically without disruption.
- **Service exposure replaces traditional systems integration**. Through WSDL and UDDI multiple services, with their own specific functionality, can be assembled to produce a system.
- **Scalability handled differently**. Registries can be used to hold lists of backup services.
- **Platform irrelevance**. Theoretically, because of SOAP standardisation, disparate platforms will interact seamlessly.
- **Federated application development**. Applications will be composed of several modular Web Services components that each provides a different piece of functionality.

For example, with the Federation Model we move towards application development that incorporates functionality aspects from multiple organisations within a group. This could open up numerous opportunities for companies to develop applications at a fraction of the effort and cost.

The shift towards such a development environment comes about because of the standards outlined within this paper. The SOAP specification forms the basis for interoperable communication. The WSDL defines interoperable Web Service descriptions. The UDDI (and other registry implementations) specification defines Web Service publishing and discovery.

These standards are currently well defined but they are still advancing. It is up to vendors to keep pace with these developments to produce suitable APIs to ensure Web Services continue their evolution. However, for now, there are also stumbling blocks to using Web Services such as performance and security, which must be addressed.

References

Bloomberg (2002). Bloomberg, J., The Seven Principles of Service-Oriented Development. Retrieved August 26, 2002, from XML & Web Services Magazine website
http://www.fawcette.com/xmlmag/2002_08/magazine/focus/jbloomberg/default.asp
Bloor Research (2002). Web Service Gotchas. Retrieved August 30, 2002 from IBM website
ftp://www6.software.ibm.com/software/developer/library/wsgotchadoc.pdf

Chappell & Jewell (2002). Chappell, D.A., & Jewell, T., Java Web Services. O'Reilly
De Jong (2002). De Jong, I., Web Services/SOAP and CORBA. Retrieved September 2, 2002, from http://www.xs4all.nl/~irmen/http://www.xs4all.nl/~irmen/comp/CORBA vs SOAP.doc
Hendricks et al. (2002). Hendricks, M., Galbraith, B., Irani, R., Milbery, J., Modi, T., Tost, A., Toussaint, A., Basha, J., Cable, S., Professional Java Web Services. Wrox
Kao (2001). Kao, J., Developer's Guide to Building XML-Based Web Services with the Java 2 Platform Enterprise Edition. Retrieved August 7, 2002, from TheServerSide.com website http://www.theserverside.com/resources/article.jsp?l=WebServices-Dev-Guide
Monson-Haefel (2002). Monson-Haefel, R., EJB2.1 Web Services: Part 1. Retrieved August 20, 2002, from TheServerSide.com website http://www.theserverside.com/resources/article.jsp?l=MonsonHaefel-Column2
Security: ebXML, SAML, XACML, WS-Security http://www.oasis-open.org/
Security: Application Security Company http://www.quadrasis.com/
Security: Liberty Alliance Security Project http://www.projectliberty.org/
SOAP 1.1 http://www.w3.org/TR/2000/NOTE-SOAP-20000508
SOAP 1.2 Messaging Framework http://www.w3.org/TR/2002/WD-soap12-part1-20020626
SOAP 1.2 Adjuncts http://www.w3.org/TR/2002/WD-soap12-part2-20020626
SOAP-DSIG, XML Encryption, XML Signature, XKMS http://www.w3.org/
SOAP Interoperability Tests http://www.apache.org/~rubys/ApacheClientInterop.html
WSDL 1.1 http://www.w3.org/TR/2001/NOTE-wsdl-20010315
UDDI 3.0 http://uddi.org/uddi-v3.00-published-20020719.htm

Appendix A - Web Service Standards Status

General

Technology Name	Ve	Release Date
Web Services Description Language (WSDL)	1.1 1.2	W3C Released 15-3-2001 W3C Working Draft 9-7-2002
Simple Object Access Protocol (SOAP)	1.1 1.2	W3C Released 8-5-2000 W3C Working Draft 26-6-2002
Universal Description Discovery & Integration (UDDI)	3.0	UDDI.org Released 19-7-2002

Security

Technology Name	Ver	Release Date
SOAP Digital Signature (SOAP-DSIG)	n/a	W3C Released 6-2-2001
XML Encryption Syntax and Processing	n/a	W3C Candidate Recommendation 2-8-2002
XML Signature Syntax and Processing	n/a	W3C Recommendation 12-2-2002
XML Key Management Specification (XKMS)	n/a	W3C Released 30-3-2002
Security Assertions Markup Language (SAML)	1.0	OASIS Standard Maturity Level 5-11-2002
XML Access Control Markup Language (XACML)	n/a	OASIS Working Draft 15, 12-7-2002
Liberty Alliance Project	1.0	Liberty Alliance 11-7-2002
WS-Security	1.0	IBM, Microsoft and VeriSign released in April 2002. Now taken over by OASIS for discussion.

Appendix B - Glossary

AXIS
: Apache eXtensible Interaction System. An open-source implementation of the SOAP specification. It is a follow on from the Apache SOAP project and represents a complete re-architecture.

ebXML
: electronic business eXtensible Markup Language. Modular suite of specifications that provides standards for message exchanges between businesses.

HTTP
: Hyper Text Transport Protocol.

J2EE
: Java based component oriented enterprise application development environment.

JAXR
: Java API for XML Registries. Provides a standard way to use different kinds of XML registries.

JAX-RPC
: Java API for XML based Remote Procedure Calls. Enables developers to build remote procedure call functionality into SOAP requests.

Liberty Alliance Project
: Single sign-on standard based on SAML which lets users who sign on to one Web Service carry over that authenticated status when moving to other Web sites. A more feature-rich second phase is expected in the near future.

OASIS
: Organisation for the Advancement of Structured Information Standards. OASIS is a non-profit, global consortium that drives the development, convergence and adoption of e-business standards.

SAML
: Secure Assertion Markup Language, Allows organisations to exchange authentication, authorisation, and profile information securely with their partners. Currently two Java specification requests (JSR 115 and JSR 155), which are associated with SAML. SAML not included as part of WS-Security which may prove a stumbling block.

SOAP
: Lightweight standard that facilitates the transport of XML data over an underlying network protocol.

SOAP-DSIG
: SOAP digital signature specifies the syntax and processing rules of a SOAP header entry to carry digital signature information.

UDDI
: Provides a framework for publishing Web Services and for the discovery & consumption of those services by interested parties.

WS-Security
: Extensions to the SOAP designed to make Web Services applications confidential and secure.

WSDL
: Provides a way to describe a Web Service in a universally understandable way.

XACML
: XML Access Control Markup Language. Expected to complement SAML. Implemented in IBM's XML Security Suite.

XKMS
: XML Key Management (submitted to the w3c in March 2001). Protocols for distributing and registering public keys, resolution/retrieval of public key, and association and retrieval of attributes in the form of 'trust assertions' with public keys.

XML
: eXtensible Markup Language. Subset of the Standardised General Markup Language.

XML Encryption
: Process for encrypting and decrypting XML documents. JSR 106 is a community request for the XML Digital encryption API. Part of WS-Security initiative.

XML Signature
: XML compliant syntax used for representing the signature of Web Services and portions of protocol messages and procedures for computing and verifying such signatures.

Extending Physical Simulation To The Audio Domain

Graham Mccann & Hugh McCabe
School of Informatics and Engineering, Institute of Technology, Blanchardstown
Dublin 15, Ireland

Abstract

Using physical simulation to control movement and interactions between objects in a 3D environment, such as a computer game, has become increasingly common However; this idea has not been extended in a general sense to the audio domain. The audio component of these systems usually comprises of pre-recorded WAV files that are triggered in response to events. We argue that this approach has serious limitations and propose instead that the physical information made available by physics engines provides an opportunity to carry out synthesis of appropriate sounds from scratch. We outline a framework for adding a sound synthesis module to a physics engine and discuss some approaches to implementing this.

Introduction

Over the past few decades the games industry has made some startling advances in the various fields that constitute it, most notably in the areas of graphics and artificial intelligence (AI). With the ever increasing capabilities of modern computer hardware, particularly processors and graphical accelerators, it has become possible to do more complex calculations on the fly, hence the current booming popularity of physical realism in computer games. A number of companies have appeared in recent years developing real-time physical simulators that can be integrated with existing graphics engines to create a visually believable virtual world. However, in order to create a truly immersive experience you need to have both vision and sound (Preece, Rogers, et al., 1994). If the sounds you hear don't match what you see, then the illusion of realism will be ruined, no matter how impressive the visual aspect may be. With the inclusion of physics engines in many modern games, the foundation is already there for the addition of sound synthesis. The physics engine already calculates a considerable amount of data for its own purposes, and would usually discard any superfluous information once it has achieved the movements required. Instead of disposing of the intermediate physical calculations we can use the data to assist in the synthesis of an accompanying effects soundtrack.

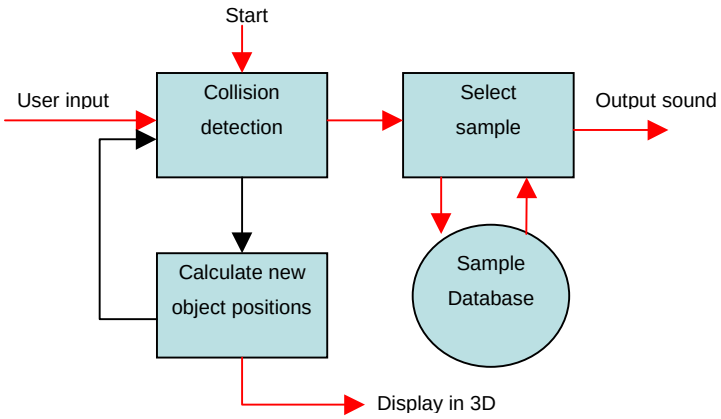


Figure 1: *A typical 3D engine for a game will use collisions between objects to trigger pre-stored samples selected from a database.*

The current trend for computer games and virtual environments is to use pre-recorded sounds, known as samples, for any of the audio components of the system. This has the advantage of requiring minimal processor time to play back the samples, since they usually can be played back without the aid of advanced filters. This method has two main disadvantages.

The first is the inconvenience involved for the developers in having to gather all the necessary sounds together before the game is completed. For large and complex games this can be quite an arduous task.

The second comes from the repetitive nature of the sounds. Due to the sounds being static recordings on the disk or in memory, each time they are played they sound exactly the same as the last time, or the last hundred times.

Computer simulated collisions, from an audio perspective at least, are a simple matter of what object collided with what, and they don't usually take into account the specifics of the collision. In reality the exact points of impact on the colliding objects are very important in determining the resultant positions after the collision. The same factors come into play when determining what sounds can be heard. Take for example, two cubes knocking together. If they were to collide face to face (i.e. flat surfaces together) you would get a particular sound. On the other hand, if one of the blocks was tilted at an angle so that one of its corners hit one of the faces of the other block, you would get a moderately different sound. The difference in sound isn't a huge amount, but enough that one would notice it. This is where sound synthesis appears to be advantageous. If the sounds were being created on the fly, then

factors such as point of impact and force of impact can be used to control the nature of the sounds being produced.

This paper outlines a new project that aims to investigate the issue of employing physically based sound synthesis in the context of a physics engine for computer games, and hence break away from the traditional method of using sampled sounds. Section 2 details the elements behind sound synthesis, beginning with it's origins in music. In Section 3 we provide a detailed discussion of modal synthesis, which is the synthesis method of choice for this project. Section 4 outlines what is involved in integrating sound synthesis with a physics engine and the final section describes future work.

History of Sound Synthesis

The idea of sound synthesis is not new; in fact it has been used since about the 1950's (Roads, 1996). What is relatively new is the idea of generating sounds in real time and from physical data, rather than just abstract parameters as it would be on an electronic keyboard for instance. With recent advances in the algorithms behind digital audio synthesis, combined with the seemingly endless increases in computer processing power and speed, it has now become possible to achieve real-time digital audio synthesis from physical data using only a regular PC.

There is an important distinction that needs to be made when discussing the history of sound synthesis; that is the distinction between analogue and digital synthesis. The first electronic device capable of digital storage didn't appear until the 1940's, however the first modern synthesiser was created at the turn of the century in 1906 by Thaddeus Cahill. It was called the Telharmonium and consisted of shafts and inductors that produced alternating currents of different audio frequencies. Unfortunately the Telharmonium weighed over 200 tons, so it was shunned as being impractical. The basic principles underlying the structure of this device were later used to great effect in the 1950's and 1960's in the form of the Hammond Organ (Roads, 1996).

The first digital synthesis experiments were conducted in 1957 by Max Matthews, et al. in Bell Telephone Labs. These experiments were a far cry from what we know today as digital sounds, and the idea of real-time synthesis was still in the distant future. In fact the calculations for the sound were carried out using powerful computers at IBM in New York and then transferred back to the Bell Labs in New Jersey, where they were played on a much less powerful machine that had audio capabilities. In the years that followed Matthews was involved in creating the musical composition languages, imaginatively titled Music, from

version WE upwards. The original Music program, written solely by Matthews in 1957, could generate only a single waveform type, only allowing the user control over the pitch and duration of each tone. Music II was written a year later for a more powerful IBM computer. This version of the program increased the number of possible waveforms to 16 and also contained four independent voices of sound. This was a big step from the previous version, but in 1960 a new development appeared in the form of the unit generator (Roads, 1996).

Unit generators are basically single components like oscillators, filters, amplifiers, etc. which can be combined together in various ways to produce synthesis instruments, also known as patches. These patches can then generate complex sound signals. The unit generators come in two types, signal generators and signal modifiers. The signal generators are what create the generic sounds in the beginning of the synthesis process. The unit generator concept was used by Matthews and John Miller that same year to develop Music III. Unlike the two previous versions, this one allowed the user to generate a large variety of synthesis techniques, thanks to the user of the unit generators. Music IV and V were developed a few years later, with version V being ported to a multitude of platforms and serving as a good introduction to digital sound synthesis for a wide audience (Roads, 1996).

Since the advent of the unit generators, and in particular their use in Music III, IV, and V, a vast array of synthesis programs and compositional languages has appeared, one of the most popular of which these days is CSound (Roads, 1996). CSound encompasses a vast variety of synthesis features, the details of which are beyond the scope of this document. However one synthesis method that it does provide which is of great interest to us is that of modal synthesis.

Modal Synthesis

Modal synthesis represents objects as collections of many smaller vibrating objects, or substructures. The fundamental theory behind it is that each of these substructures has a set of natural modes of vibration, which are exhibited whenever the structure becomes excited; in our case this would be due to collisions between objects, but it can also occur due to sliding, scraping, pressures, etc. In modal synthesis these mini-objects are modelled mathematically as a set of modal data, consisting of the frequencies and damping coefficients. This data can either be obtained through already documented experiments in engineering, or experimentally through the use of existing equipment in the industry, such as the Active Measurement Facility (ACME) at the University of British Columbia, which has the capability to automatically acquire sound measurements by moving a sound effector around the surface of a test object by a robot arm (Doel, Pai, 2001).

A good physical system which can be used as a base for creating a modal synthesis system is one consisting of a mass and a spring attached to a static surface that can be used to represent the miniature objects that make up the complete object that is being modelled (Cook, 2002). Considering that in acoustics the sounds that are produced are caused by vibrating objects, the use of masses and springs to model them is a logical correlation. If, in one of these systems, the mass is pulled (or pushed) in a particular direction, then the spring will attempt to restore the system to its rest state, initially by applying a force in the opposite direction to the original movement. While the system is attempting to return to rest, it will behave as a simple harmonic oscillator.

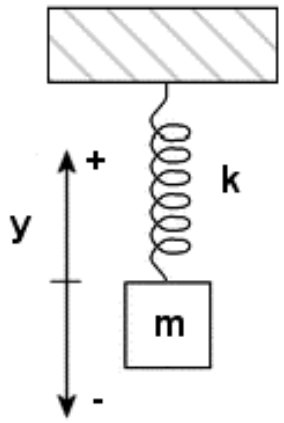


Figure 2: Mass-spring system

In each of these mass/spring systems, the mass is denoted by the symbol, *m*. The spring is defined as being the force required to displace the spring from its rest position, with the force per unit distance being represented as *k*. As with any physical system, energy losses are a factor. Here they are referred to collectively as damping, and denoted by the symbol, *r*. In most systems the damping takes the form of forces such as air resistance, gravity, friction, etc. To use a musical analogy, on a piano damping is imposed through the use of felt pads which deaden the sound after a key is released by the player. The displacement of the mass from rest position is denoted by *y*. Using the same variables, we can now apply Newton's second law of motion to the system to get the following equation:

$$-ky - mg - rv = ma$$ (Force = mass x acceleration)

Here we also introduce the new variables for gravity, *g*, velocity, *v*, and acceleration, *a*. Compared to the magnitude of the force exerted by the deformed spring, the force exerted by gravity is negligible, so we can simplify the equation by removing this operand. This leaves us with:

$$-ky - rv = ma$$

In order to get a solvable equation we need to make some substitutions. Knowing that velocity is the rate of change of position *y* with respect to time, *t*, and acceleration is the rate of change of velocity with time we can get the following:

$$-ky - r\frac{dy}{dt} = m\frac{d^2y}{dt^2} \quad \text{or}$$

$$\frac{d^2y}{dt^2} + \left(\frac{r}{m}\right)\frac{dy}{dt} + \left(\frac{k}{m}\right)y = 0$$

This equation has a solution of the form:

$$y(t) = y_0 e^{(-rt/2m)} \cos\left(t\sqrt{k/m - \left(r/2m\right)^2}\right)$$

This basically informs us that the mass will oscillate up and down in simple harmonic motion, i.e. with a constant frequency. Having to model one of these systems for each mode of each component of an object would be quite an excessive load on the processor for a real-time system. One way of decreasing this load is to use digital filters. We can quantise the previous equation using the following approximations:

$$dy/dt = (y(n) - y(n-1))/T$$
$$d^2y/dt^2 = dv/dt$$
$$= (y(n) - 2y(n-1) + y(n-2))/T$$

where *n* is the sampling number and *T* is the sampling interval. This would yield an equation of the form:

$$y(n) = \frac{y(n-1)(2m+Tr)}{(m+Tt+T^2k)} - \frac{y(n-2)m}{(m+Tr+T^2k)}$$

Although this looks drastically different to the original continuous equation it can be shown experimentally that the two solutions yield results that are quite similar (Cook, 2002). Having the equation in this digital form allows it to be implemented through use of digital electronics like resonant filters set to the previously mentioned resonance frequencies, or in our case simplified algorithms for a computer program that simulates these same filters.

However knowing how to solve a single mass/spring system or having a single modal filter is only a small part in being able to physically model an object. We need to form a type of mesh to represent the object in all dimensions, and to do this it will be necessary to have multiple masses connected together by multiple springs, or the equivalent using digital electronics (Cook, 2002). This not only makes the mathematical calculations more complicated, but it also increases the processing power and time that will be required for a real-time system to able to keep up with the graphical simulation. Once the waveforms have been calculated for each sub-component, they are combined using basic sinusoidal waveform addition.

Integrating Sound Synthesis with a Physical Simulation

Previous work has been done in the area of real-time sound synthesis along with the use of a custom built graphics application (Doel, Kry, Pai, 2001). This project aims to take a more generic approach to allow integration with virtually any existing physical simulation engine available. So before discussing how the sound synthesis is integrated with an existing engine, it is prudent to discuss briefly what a physics engine is, and does. The primary purpose of using a physics engine in an application is to include as much physical realism as possible. Instead of using the traditional methods of animation, like predetermined paths for objects, we can now simply model the virtual world just as one would the real thing, and then let the physics take over. Natural forces like gravity and friction exist in the system as standard, but it is also possible to introduce user-created forces to influence and control the system. The most common feature used in these engines is that of rigid body dynamics.

While it doesn't do all the work for us, it does relieve much of the workload by doing all of the complex physical calculations. In general the physics engine does not provide any additional features other than the physical data, so it is left up to the user to create the graphical engine and, in particular for this project, the sound engine. With the existence of modern hardware-independent graphics APIs, like OpenGL and DirectX, the creation of a graphical display system for an application is not the mammoth task it once was.

As such, the Havok physics engine is one of the key elements of this project, providing real-time physical simulations where required. In recent times the popularity of Havok technology has gone from strength to strength, with major graphics companies incorporating it into their projects, for instance Discreet. Macromedia and Intel have also chosen the Havok system as the backbone to their Shockwave 3D platform (Havok, 2002).

For the purposes of demonstrating the methods researched throughout this project a synthesis engine is being developed. This engine is being developed in three stages: the first being a basic traditional sound engine, i.e. one which uses sampled sounds, rather than synthesising them in real-time. The purpose of this part is to become familiarised with the operation of such an engine when used in conjunction with the Havok physics engine, or any other physical simulator for that matter. At this point in the project, this stage in the development has already been completed. The second stage, which is currently underway, involves the actual synthesis side of the project.

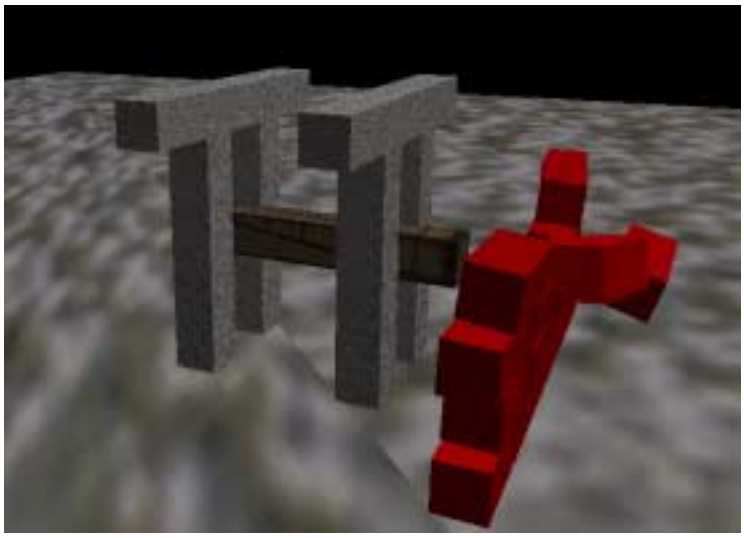


Figure 3: An early demo using the Havok physics engine

In the current state of development of the project a synthesis API developed by Perry Cook, known as the Synthesis Toolkit in C++ (STK) [REF], is being used to demonstrate the possibilities of having a full sound synthesis engine. The toolkit is a set of digital audio signal processing and synthesis classes which can be integrated into an existing program to allow the generating of synthesised sounds. Initially, the STK is being triggered by any collision and plays a synthesised sound according to the material specified in the demo. As the development progresses this will be upgraded to incorporate the detailed physical simulation that already exists in the graphical components. The STK classes contain methods by which we can use the algorithms necessary to create sounds through modal synthesis. These methods take a number of parameters, such as magnitude of impact force, points of impact on the colliding bodies and shape and composition of the objects, and then calculates the appropriate sounds.

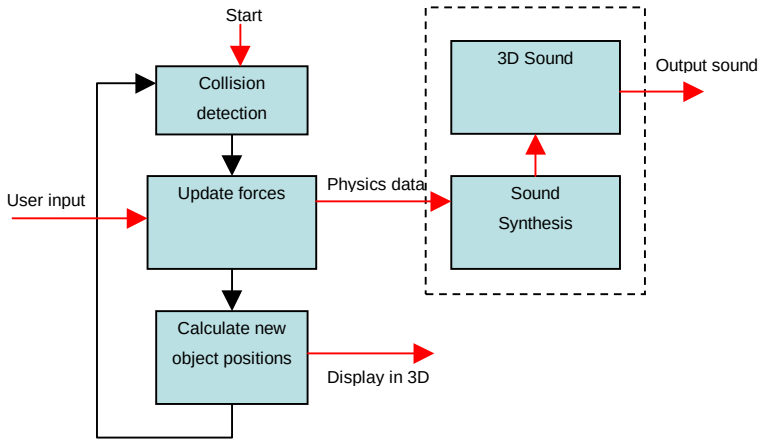


Figure 4: Synthesising sounds based on data from physics engine

Future Work

The next major step in the development will be to use the data retrieved from the Havok engine to more accurately control the output from the synthesis engine. Whether or not modal synthesis is kept as the method of choice for this project remains to be seen, but either way a module will need to be developed that will interpret the physical collision information outputted from the Havok engine and translate it into a format that can be used as input for the synthesis engine.

As it stands, all the research done up until now suggests that modal synthesis is probably going to be the most effective synthesis method. However, as more detailed research is carried out into the inner workings of the various methods available, it may become evident that some of the other synthesis techniques might be more appropriate, or even that it may be more appropriate to use different techniques for different types of objects.

In graphical modelling there is a concept known as level of detail, whereby the model only displays enough detail that the user will notice. We can use a similar technique to reduce the level of detail of the sonic model, by decreasing the number of masses (and hence the number of springs, and thus calculations) in the body. Incorporating this feature will allow us to find a good balance between performance and audio adequacy. Depending on the computers capabilities, or simply the number of sounds that need to be played at the same time, the amount of processing can be drastically reduced for each sound by reducing the number of modes calculated. Also, by giving priority to the higher amplitude modes, we can make sure that the sounds still bear a close resemblance to the real thing, even with only a few basic modes, rather than the full range. This is possible because when all the modes are resonated together the lesser ones are somewhat drowned out by the others. Although having all of them play together gives it a much fuller sound, it isn't necessary to be convincing.

Conclusion and Comments

This paper has described an on-going research project that aims to incorporate a form of physically generated sound synthesis into an existing game engine. We have outlined a framework for incorporating this into a physics engine and pointed out the advantages of implementing this approach. We concluded by detailing the workings of modal synthesis, which has been determined as the most appropriate method for these purposes, and by examining some of the existing work done within the graphics community in this area.

At present a number of demonstration applications have been developed as well as some preliminary research into various synthesis techniques. In the immediate future the intention is to continue more detailed research into modal synthesis and how best to implement a generic sound engine that can be integrated with the existing or future physics engines, and extract any necessary physical data from them.

References

Cook (2000). Perry R. Cook, *Physically-based Parametric Sound Synthesis and Control*, In Proceedings of SIGGRAPH 2000 Conference.

Cook (2002). Perry R. Cook, *Real Sound Synthesis for Interactive Applications*, A K Peters, Ltd., Natick, MA 01760.

Doel (2001). Kees van den Doel, *Modal Synthesis for Resonating Objects*, To appear in a book. Available from: `http://www.cs.ubc.ca/~kvdoel/`

Doel, Kry, Pai (2001). Kees van den Doel, Paul G. Kry, and Dinesh K. Pai, *FOLEYAUTOMATIC: Physically-based Sound Effects for Interactive Simulation and Animation*, In Proceedings of SIGGRAPH 2001 Conference.

Doel, Pai (1996). Kees van den Doel and Dinesh K. Pai, *The Sounds of Physical Shapes*,

Doel, Pai (2001). Kees van den Doel and Dinesh K. Pai, *Modal Synthesis for Resonating Objects*,

Havok (2002). Havok, Retrieved on October 12 2002 from: http://www.havok.com.

O'Brien, Cook, Essl (2001). James F. O'Brien, Perry R. Cook, and Georg Essl, *Synnthesizing Sounds from Physically Based Motion*, In Proceedings of SIGGRAPH 2001 Conference.

Preece, Rogers, et al. (1994). J. Preece, Y. Rogers, et al., *Human-Computer Interaction*, Addison Wesley Publishing Company.

Roads (1996). Curtis Roads, *The Computer Music Tutorial*, The MIT Press, Cambridge.

Young, Freedman (1996). Hugh D. Young and Roger A. Freedman, *University Physics*, Addison Wesley Publishing Company, Inc.

Takala, Hahn (1992). Tapio Takala and James Hahn, *Sound Rendering*, In Proceedings of SIGGRAPH 1992 Conference.

Smith (1992). Julius O. Smith III, *Viewpoints on the History of Digital Synthesis*, In Proceedings of the International Computer Music Conference, Montreal.

Hardware and Software Codesign for Multimedia Capable Portable Devices using SystemC

Richard Gallery Deepesh M. Shakya

Institute of Technology, Blanchardstown
Blanchardstown, Dublin
email: {Richard.Gallery, Deepesh.Shakya}@itb.ie

Abstract

Multimedia capable portable devices such as 3G phones will host a variety of new applications. Although the underlying push for new applications in such devices is driven by the increase in bandwidth offered by 3G, it is clear that many of the "new" applications will require the provision of new and powerful graphics/video technology within the mobile device itself. Within a computing device, high bandwidth and computational cost are associated with anything but the simplest of graphics, and as a result the graphics subsystem is generally one of the most critical elements of a system, requiring particular attention in the design process. The project is examining the suitability of SystemC, a system description language, for Hardware/Software Codesign of a graphics system in a typical next generation WAP compatible device.

1. Introduction

Multimedia capable portable devices, one early example of which is the WAP compatible mobile phone, incorporate a combination of computing, wireless communications, signal processing, graphics and other technologies to provide a cost effective solution to the customer. Custom designed IC's are normally required in order to implement these demanding technologies in a cost-effective manner. Increasingly such IC's are Systems-On-A-Chip, where a variety of digital, analogue and software technologies are integrated together in the interests of cost reduction.

Hardware/software co-design plays a vital role throughout such development projects, especially during the initial stages where critical decisions regarding hardware software partitioning and system constraints are taken. However, tools to allow hardware/software co-design are not yet fully developed. In a typical hardware/software codesign, the systems level engineers would model and program in C/C++, whereas the hardware engineers would design using a hardware description language such as VHDL, leading to communication and other difficulties within a project team. Open SystemC is an industry led initiative which addresses this issue by seeking to establish a modeling platform that promotes and accelerates system

models using standard ANSI C++ with SystemC extensions (in the form of classes and data structures), which can then be proven using simulation tools before transfer to silicon.

2. Next Generation Graphics/Video requirements for Multimedia Capable Portable Devices

The next generation of mobile telecommunications is capable of providing data rates of up to 2 Megabits per second 0. This offers the prospect of broadband video and multimedia services on the move, such as Video Conferencing, on-line entertainment and Internet access. For all these features, graphics/video system of the mobile system should be designed with appropriate functionality.

A part of the project deals in the study of graphics/video requirements for next generation mobile phones. Some of the applications to be made available in next generation mobile phones include Video Conferencing, Video Streaming, Multimedia Messaging Service, Gaming, Integrated Digital Camera and Camcorder, Video Clips play back etc. In order to support these applications, the devices must be enabled with appropriate graphics/video functionalities. For e.g. Video Conferencing and Video Streaming demand good image quality. To ensure this an appropriate encoding/decoding standards must be adopted.

To support fully fledged 3D gaming, the device must be enhanced with a 3D graphics engine which supports perspective correct texture mapping, bilinear, MIP-mapping, Gouraud Shading, alpha-blending, Stippling, anti-aliasing, fogging and Z-buffering.

The device must also support other graphics functionalities like scaling, scrolling, vertical and horizontal filtering, multiple video overlays etc. Also, the device must be equipped with image grabbing functionality to enable it serve as a digital camera or camcorder.

3. An Overview of video mixing

Graphics/video systems are often capable of generating overlayed or composite images (which enables menus, graphic overlays, picture in picture etc.). The video/graphics subsystem capable of performing this task is usually termed the mixer. The simulation of a mixer is carried out as an initial aspect of the overall project to gain familiarity with the issues arising in graphics/video systems.

Mixing in a video/graphics system may be described as the combination of two images in which one image (the overlay image) is overlayed on another image (the primary image) according to an *alpha* value[4].

For each pixel in an image the output pixel that results from the mixer is governed by the following expression:

$$\text{pixel}_{\text{output image}} = \alpha \; x \; \text{pixel}_{\text{primary image}} + (1.0 - \alpha) \; x \; \text{pixel}_{\text{overlay image}}$$

Normalisation of the output value is assured through the use of *alpha* and (*1.0-alpha*). The value of alpha varies from 0.0 and 1.0.

The *alpha* value may be fixed for the entire image, in which case the images are combined in a similar manner at each pixel. In practice a system that can only achieve this would be of little value. A more useful mixer results if the alpha value may be allowed to vary such that there is a separate alpha value at each pixel. This approach allows the manner in which images are combined to be varied from one pixel to the next. Other approaches could involve fixing the alpha value for some portions of the image, and varying it for others.

Although the concept above is illustrated using floating-point calculations, in practice floating point arithmetic is not required to perform these calculations. Instead the *alpha* values may be represented as integers in the range 0..255[5]. In this case integer arithmetic may be used. Each multiplication of 8 bit numbers will produce a 16-bit result, but the lower 8 bits may be disregarded, to produce an 8-bit output.

$$\text{pixel}_{\text{output image}} = \alpha \; x \; \text{pixel}_{\text{primary image}} + (255 - \alpha) \; x \; \text{pixel}_{\text{overlay image}}$$

Figure 3-3 shows the output image obtained after mixing images in Figure 3-1 and Figure 3-2. The value of alpha taken for this mixing operation is 127 (applied across the entire image).

[4] In our research a base image format of RGB 8:8:8 with an additional 8 bit alpha value is assumed.

[5] Or other ranges if appropriate to the system requirements, for example some systems might find a more restricted alpha range sufficient.

Figure 3-1 Primary Image

Figure 3-2 Overlay Image

In order to gain familiarity with the techniques involved, an algorithmic simulation of the mixer was carried out. This algorithmic simulation is done without taking into account many of the hardware aspects required by a real mixer, and serves to provide a general understanding of the working mechanisms of the mixer. In addition the results obtained from the algorithmic simulation can be used to provide a test bench against which further, more involved, simulations can be verified[6].

Figure 3-3 Mixed Output Image

Once the algorithmic simulation has been developed to a sufficient level a hardware simulation can also be developed. Thus the mixer is developed for both algorithmic simulation and hardware simulation.

[6] An advantage of working with both the hardware and algorithmic simulations at the same time is that the output obtained from the hardware simulation can be compared with the output obtained from the algorithmic simulation. The results obtained from both simulations should be same. To check this, a test class is developed to compare the output of two simulations. This can be done by comparing the corresponding pixels for both the outputs images. If they are all the same then there are no errors in the hardware implementation.

4 System Architecture for the Graphics/Video Subsystem

Our initial hardware simulation of the mixer serves to illustrate the manner in which hardware simulation may be carried out in a programming language like C++. Amongst issues that arise in the hardware simulation in C++ are those of synchronisation and parallelism, for which there is no inherent support in C++. These are some of the areas in which System C[7] offers a solution for the development of hardware simulations in a high level language.

The image data for the overlay image and the principal image, alpha value and output image data are stored in the main system memory (RAM). These data are transferred into the mixer. The two images that are subjected for mixing operation have been named as the overlay image and the primary image here. The overlay image is overlayed on the principal image.

Rather than mixing one pixel at a time the images are mixed an image block at a time (e.g. 8 pixels at a time, or 64 pixels at a time). Data is transferred from the RAM to the mixer in data blocks. The minimum size of a data block depends upon the size of the data bus (for example, a 64 bit data bus would allow a minimum size of 8 bytes for data transfer blocks. An image block will consist of one or more data blocks. For example, the alpha image has one data block (the 8 bit alpha values) per image block, whereas the primary image has three data blocks per image block. The three data block accounts for the Red, Green and the Blue components of the primary image.

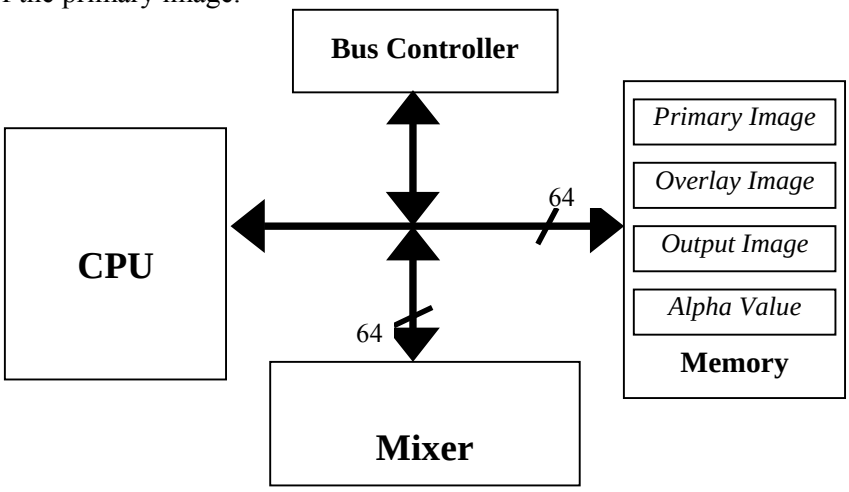


Figure 4-1 System Architecture of the Mixer

7 SystemC is a modeling platform consisting of C++ class libraries and a simulation kernel for design at the system-behavioral and register-transfer-levels.

The data bus may be occupied by memory transfer associated with devices other than the mixer (for example there may be a processor on the bus), so prior to the transfer of data into the mixer, the mixer should take control of the data bus (become bus master). The top-level system architecture of the hardware simulation of the mixer is shown in Figure 4-1.

4.1 Mixer Library

The classes that have been used for the hardware simulation of the mixer are categorised as follows:

Utility

Utility classes consist of the classes relating to the allocation of memory, generation of alpha values, reading and writing bitmap images, conversion of image data from its interleaved[8] form to the deinterleaved[9] form and vice versa.

Interface Classes

A number of interface classes have been defined in order to link two different classes that may be a class from the Hardware Simulation classes and a class from the Utility classes. If a class in the Hardware Simulation classes needs information from the Utility classes, it shouldn't receive this information directly. It should get it through an interface class. This makes the classes completely independent of each other.

Hardware Simulation

Various classes have been defined to simulate the different hardware aspects of the mixer.

Algorithmic Simulation

A separate class has been defined for the algorithmic simulation. This class performs mixing operation irrespective of the hardware contents. The main objective of this class is to generate the reference output data to check the correctness of the output obtained from the hardware simulation.

Test Routines

It is necessary to check the correctness of the output obtained from the hardware simulation of the mixer. Therefore, a class has been created that takes the output data

[8] Image data with all image components i.e. red, green and blue.

obtained from the hardware and algorithmic simulation of the mixer and then compares these two outputs.

5 RTL Hardware Model of the Mixer

Having performed an algorithmic simulation, followed by a high level architectural simulation, the mixer was next simulated in C++ at the RTL[10] level. Here, the mixer hardware is controlled by a micro-programmed control unit [8], where the micro-program consists of a series of instructions that correspond to the control signals (for the hardware) required to carry out the mixing operation.

5.1 System Architecture

The architecture of the hardware is designed on the principle that registers are connected to other registers via gates. In practice a real hardware implementation would be implemented slightly differently, gates would be connected via a bus and data would be transferred between them by latch enabling the register (that the data would be transferred to). The gating model introduces an additional element to achieve the same effect, but it is a conceptual abstraction, which may be useful in the modeling of the hardware, without adding any additional requirements to the hardware implementation [9].

The model presented in this document consists of three main units: the Control Unit (CU), the Data Addressing Unit (DAU) and the Arithmetic Unit (AU) of the mixer. The CU generates a series of control signals, which are provided to the DAU and the AU of the mixer, and dictate the tasks to be carried out in these modules.

Figure 5-1 shows a high-level system architectural model of the hardware mixer.

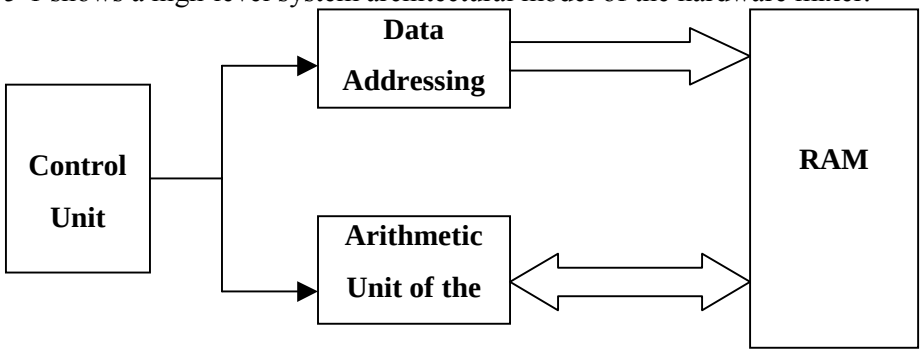


Figure 5-1 System Architecture of RTL Hardware model of the Mixer

[9] Image data that consists of specific component only red, green or blue.
[10] Register Transfer Level. In the RTL level model, the complete system state is separated into groups of bits (called registers) and considers the flow of information from one register to the next in each clock cycle.

The DAU generates the addresses of the data (to be fetched from the memory) as indicated by the microprogram. The AU on the other hand carries out the mixing operation (after receiving the required data i.e. primary image data, overlay image data and alpha value).

The CU consists of an Instruction Store (IS), Instruction Register (IR) and the Program Counter (PC). All instructions are stored in the IS. Each (microprogram) instruction consists of the following: -

- Control signals, to control the hardware
- A flag (BR) to indicate whether the instruction is branchable or not
- A field (BRT) to indicate the type of branching
- The branch address, if any

The output of the CU is the control signals required for each instruction to be executed. Control signals are connected to the controlling inputs on the hardware models for e.g. a gate is opened only when the control signal connected to it is enabled. This is shown in Figure 5-2.

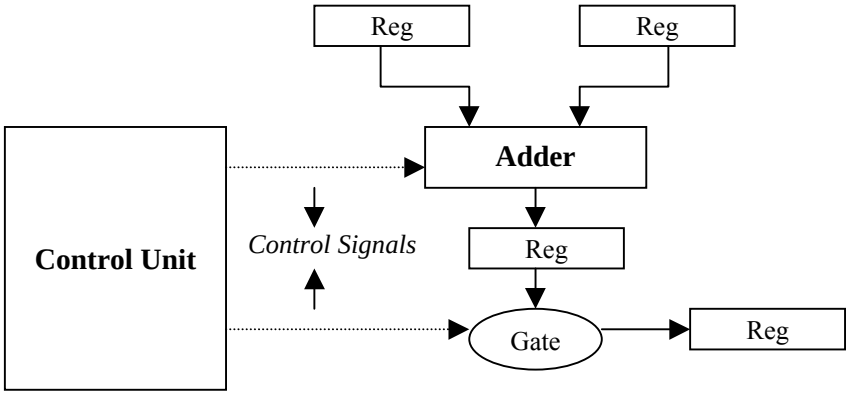


Figure 5-2 Control of an Adder and a Gate through control signals

5.2 Branching in the CU under direction of micro-instructions and signals

Execution of the instructions stored in the IS begins with the initialization of the PC with the address (within the IS) of the initial instruction. The instruction corresponding to the address contained in the PC is loaded from the IS to the IR. If the BR flag is not set, the PC is incremented i.e. ready for the next instruction in the IS to be fetched. If the BR flag is set, the type of branching is first checked. If the branch is of type BR_JUMP_COUNT_NZ, then the input control signal IP_COUNT_Z is checked. If this signal is not set then the next instruction will be the instruction contained in the address of the previous instruction and this

is loaded into the PC. If the input control signal IP_COUNT_Z is set, which means all the pixels have been processed, the program counter is incremented by one. The next instruction is I_STOP, which would terminate the mixing process. If the branch is of type BR_NO_BRANCH, then the PC is incremented.

Figure 5-3 shows the flowchart of the pixels being processed.

Figure 5-4 gives a model of the CU.

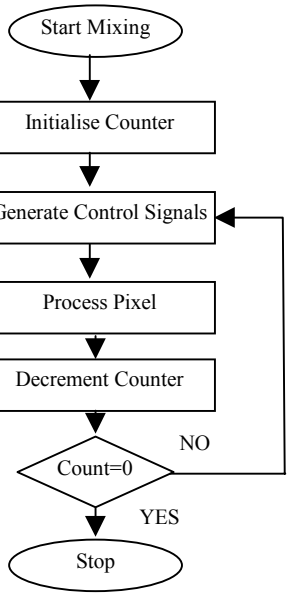


Figure 5-3 Flowchart of the mixer operation

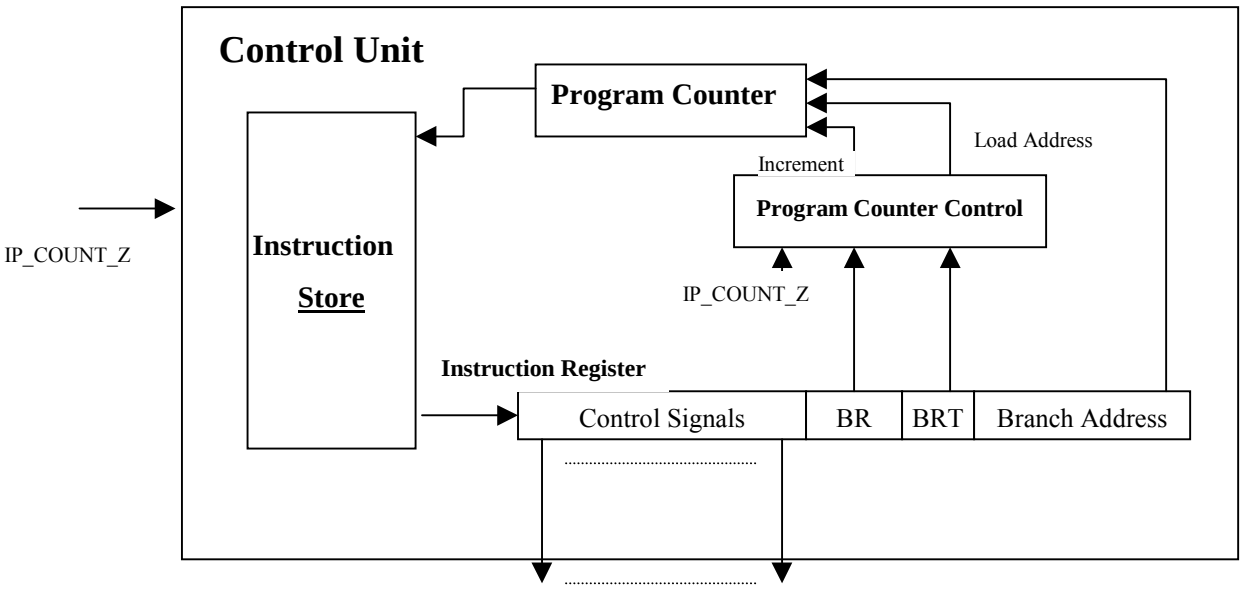


Figure 5-4 The Control unit of the mixer

5.3 Conclusion on this Hardware Model

This approach worked as expected. Two bitmap images of size 640x480 were mixed to get mixed image output. The output obtained was compared with the output obtained from the algorithmic simulation. The corresponding RGB components for every corresponding pixel of the two output images were subtracted, and the result obtained for all the pixels was found to be zero, indicating the desired output has been obtained.

Additionally this model also supports synchronisation and parallelism. For this a hardware specific simulation kernel has been developed. In each cycle, an instruction is executed and then all the registers are updated.

6. SystemC

SystemC [1][2][3] is a modeling platform consisting of C++ class libraries and a simulation kernel for design at the system-behavioral and register-transfer-levels.

SoC (system on chip) designs are a combination of hardware and software - not just hardware only as in ASIC design. In fact, there is more software than hardware in most designs. Therefore, there is a need for a language that describes both the functionality of the software and the hardware. SystemC is one solution, which satisfies this requirement.

SystemC has different features to assist in the system level design. It consists of *Modules.* These are the basic building blocks for partitioning a design. They allow a design to be separated into more manageable pieces and to hide internal data representation and algorithms from the other modules. A typical module consists of *ports* for the module to communicate with the environment, *processes* that describe the functionality of the module, *internal data* and *channels* for communication among the module's processes. The design, which maintains hierarchy, contains modules within modules. The other features in SystemC include a rich set of signal types, data types, clocks, reactivity, multiple abstraction levels, functional models, fixed-point data types, and communication protocols [1].

7. Hardware Model of the Mixer using SystemC

The problem in C++ is that it cannot easily be used to describe hardware as it doesn't have a natural way to represent constrained data types, concurrency and clocks 0. This problem is

solved using SystemC. SystemC has tools such as concurrency, reactivity[11], data-types required for modeling the hardware. SystemC also supports hierarchy of the modules. The concept of modules in SystemC allows us to build separate entities[12] and the communication between these entities is carried out through channels. The channel we have used in the hardware model of the mixer is *sc_signal<T>*. This channel implements both the *in*- and *inout*-interfaces[13].

In the current implementation of the mixer in SystemC, a separate module has been created for each entity for e.g. registers, adder, multiplexer, gates of the DAU and subtractor, adder of the AU. A better approach would have been to maintain the hierarchy of modules by constructing modules for the DAU, AU and CU. Then building different modules (for e.g. adder, register, multiplexer, gates etc) inside these modules. This approach will be taken in later simulation process of the project.

One advantage in SystemC is that it allows the user to make use of a user defined *Packet Type* which is similar to the data type. The input and output ports of the modules and channel (*sc_signal*<T> in our case) could be defined as the packet type. The packet type is defined by a structure. SystemC allows a user to pass this packet type from module to module. The advantage of the packet type is that an entire structure of data can be transferred from one module to another through a single port.

The control unit (CU) part of the mixer consists of an instruction store (IS) and the program counter (PC). The initial task in designing the CU for the mixer was to define the packet type for the instruction. This packet type consists of the information on the instruction type ID, branch type, branch address and the array of control signals generated by the instruction, as shown below.

```
struct instruction_type
{
        instruction_type_E ID;
        branch_type_E br_typ;
        int branchable;
        int br_addr;
        int op_ctrl_lines[C_NO_CONTROL_SIGNALS];
        ……..
};
```

[11] Hardware can be taken as a set of non-terminating process that reacts continuously to events in their environment 0.
[12] Entity in this document means the hardware abstraction of the digital system.
[13] An interface that lets data in and out of the module through its ports.

As it is seen in the above code, the control signals have been defined as *int* type. In fact, it should be *bool* type. Due to some reasons, the simulation was not working properly with the *bool* type. This problem will be resolved as we take further steps in SystemC simulation process.

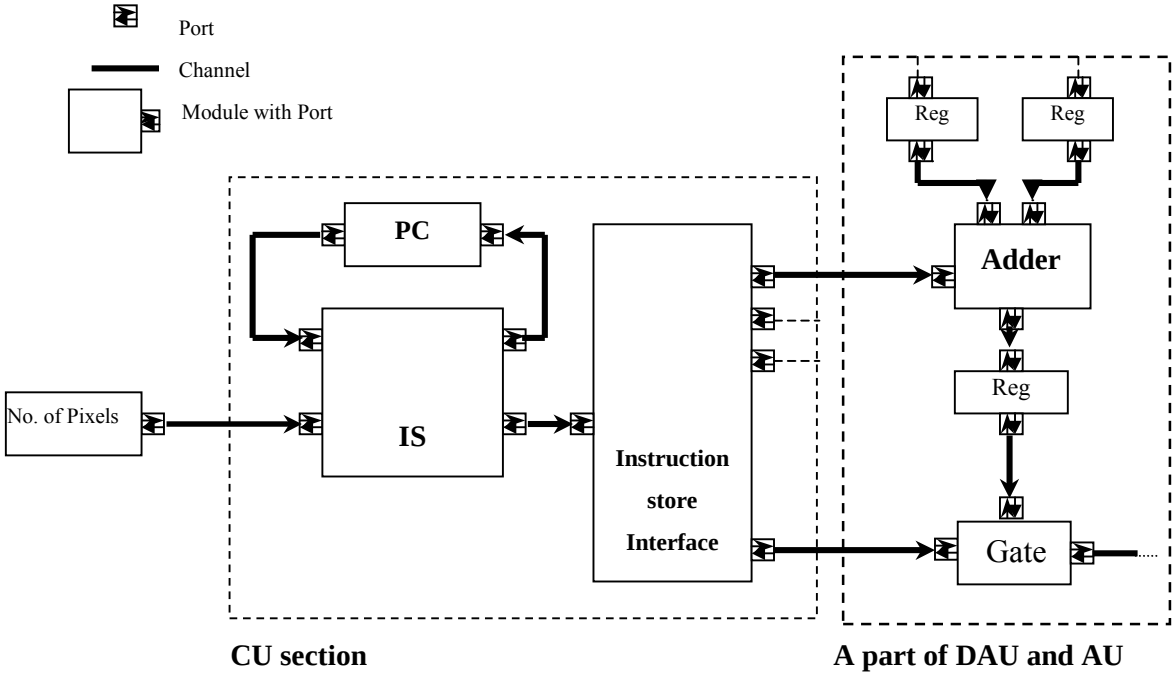


Figure 7-1 Hardware Mixer Model in SystemC

The next step was to define a module for the IS. The IS holds the microprogram that is required to control the hardware to carry out the mixing operation. This module consists of two output ports and three input ports. The first output port is defined as the packet type for instructions. The second output port is for delivering the address of the next instruction if the current instruction is *branchable*. The first input port takes the input from the PC that informs the instruction store which instruction to fetch. The second input port takes information of the number of lines processed. This is shown in Figure 7-1. The third input port is for the clock. The port for the clock is not shown in the figure. When all the lines in the image have been processed then the IS module ceases operation.

The ports for the IS module are defined as follows: -

```
sc_out<instruction_type> inst_out_opin;
sc_out<int> next_address_opin; //to be supplied to the PC

sc_in<int> PC_ipin;  //program counter input
sc_in<int> no_lines_scanned_ipin;
sc_in<bool> clk;
```

In the code given above *instruction_type* is the packet type. The IS module outputs the instruction to its output port as the packet type. The control signals are embedded inside this packet type.

The input ports of the other modules in the mixer (for e.g. gates, adder etc) take control signals as the input (Figure 7-1). Thus, to establish communication between the IS module and the other modules in the mixer, the input port of the latter module should be of the same type as the output port of the former module. For this a separate module is created called an *instruction interface* module. This module takes input of the packet type and delivers separate outputs for each control signals. Thus an interface is required between the IS module and the other modules to establish proper communication between them. In Figure 7-1, the IS and the instruction store interface have been shown as separate modules. The better approach would have been to define these modules in a hierarchical manner by having these modules defined inside another module. This module would then act as the IS module. As was said earlier, the module hierarchy approach has not been observed in this SystemC module. This will be corrected later in the project.

```
SC_MODULE (instruction_interface_mod)
{
        sc_in<instruction_type> inst_str_ipin; // packet type input

        sc_out<int> ID_V;             // defined only for debugging purpose
        sc_out<int> br_typ_V;      // not required for interface
        sc_out<int> branchable_V;
        sc_out<int> br_addr_V;

        sc_out<int> C_INIT_REG_BASE_ADD_ALPHA_V;  //separate output for
        sc_out<int> C_INIT_REG_BASE_ADD_RED1_V;  //each control signals
        ……;
}
```

The PC module takes input from the IS module. If the current instruction is not *branchable* then the PC is simply incremented. If the current instruction is *branchable* then the IS module provides the PC with the address of the next instruction to be fetched.

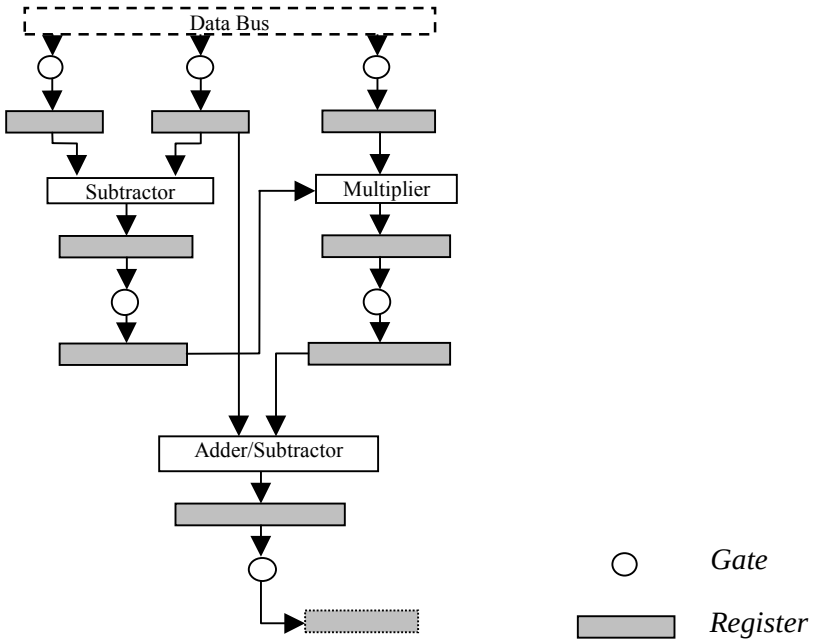


Figure 7-2 Arithmetic Unit organization

Having implemented the CU, the next step was to create modules for different components of DAU and AU.

The modules for the DAU are multiplexer, adder, registers, multiplier and gates. The modules for the AU are subtractor, adder/subtractor, multiplier, registers and gates. The organization of the AU is shown in Figure 7-2. Each module is provided with its own sensitivity list[14]. A module is activated if an event occurs in any one of the members of its sensitivity list. Registers are made sensitive to events in the input port. Whenever there is a change in a register input then the input of the register is transferred to its output. Gates are made sensitive to the control signals which come with the instructions. Other modules such as adder, subtractor, multiplier, and adder/subtractor are also made sensitive to the control signals (Figure 7-1). These modules are triggered only when they are told to do so by the instruction.

7.1 Conclusion on SystemC approach

SystemC provides the tools required for the hardware modeling. The provision of modules and ports support hierarchy of modules whereas the provision of channels provides an

[14] Sensitivity is a list of variables for a module. If event occurs in any one of them, the module reports corresponding changes. The module will not invoke unless an event occurs in the member of the sensitivity list.

abstraction for communication. These features are not available in C++ based hardware modeling.

The model is working as expected. The method of testing the outputs in this approach is not as straightforward as in the earlier approach. The outputs are first stored in the file. The output data in the file is later on compared with the output obtained from the algorithmic simulation. There might be some efficient way of testing but it remains as a work to be explored in the later part of the project.

8 Further work

Having completed the current work, a system requirements specification will be produced for a graphics/video system.

A graphics/video system will be designed that complies with the SRS earlier generated for the graphics/video system. The graphics/video system will then be implemented using SystemC.

After completing this, a research will be conducted into the hardware/software partitioning of the hardware/software Codesign and techniques for facilitating this within SystemC. A report will be produced highlighting the suitability of SystemC for the design.

References

[1] Joachim Gerlach, Wolfgang Rosenstiel, *System Level Design Using the SystemC Modeling Platform,* University of Tübingen, Germany, Workshop on System Design Automation (SDA'00), Rathen, Germany, March 2000, pp. 185-189.

[2] Thorsten Grotker, Stan Liao, Grant Martin, Stuart Swan, *System Design with SystemC,* Design, Automation, and Test in Europe (DATE '01) **.**

[3] *SystemC Version 2.0, User's Guide,* www. SystemC.org.

[4] Stuart Swan, *An introduction to System Level Modeling SystemC 2.0,* Cadence Design Systems, Inc. May 2001.

[5] *Official Website of Nokia mobile phones,* www. nokia.com.

[6] G. Economakos, P. Oikonomakos, I. Panagopoulos, I. Poulakis, and G. Papakonstantinou,

[7] *Behavioral Synthesis with SystemC,* Design, Automation, and Test in Europe (DATE '01)**.**

[8] William Stallings, *Computer Organization and Architecture*, pp. 578-617, Prentice Hall.

[9] William Stallings, *Computer Organization and Architecture*, pp. 554-576, Prentice Hall.

[10] Dr. Guido Arnout, *SystemC Standard,* Proceedings on the 2000 conference on Asia and South Pacific design automation January 2000.
Stan Y. Liao, *Towards a new standard in System Level Design*, Advanced Technology Group, Synopsys, Inc.

Unified Messaging Systems: An Evolutionary Overview

Declan Barber and Anthony Keane,

Institute of Technology, Blanchardstown

declan.barber@itb.ie

anthony. keane@itb.ie

Abstract

Over the last decade, the widespread demand and use of the internet has changed the direction of the telecommunications industry as it was recognised that the internet could be used as an inexpensive way to handle not only data but also voice communications. This convergence of traditional voice and data technologies towards an IP-based open architecture has been paralleled by a convergence of the internet and mobile communications. As a result of these convergences, unified messaging has emerged as a technically viable service. Integrated messaging services that offer partial unification of different message types are already in the marketplace. This paper asks what unified messaging means and examines underlying architectural developments that are likely to shape the unified messaging applications of the future.

1 Introduction

Traditionally, circuit switched voice networks and packet-switched data networks have been kept separate in the enterprise (see Figure 1). During the 1990's, however, the widespread use of the internet changed the direction of the telecommunications industry as it slowly recognised the potential of the Internet as an inexpensive way to handle not only data but also voice communications. The ubiquitous Internet Protocol (IP), a simple and effective protocol for packet delivery, is independent of the information it carries and with its open architecture, easily inter-operates with other protocols. Although challenges remain to achieve network convergence, the promise of communications that offer the quality and reliability of traditional voice systems with the efficiencies and manageability of TCP/IP over an enterprise network infrastructure is highly attractive to the enterprise. This convergence between traditional telephony and the internet has been paralleled by the convergence of internet with mobile technologies. One of the outcomes of these convergences has been the emergence of unified messaging (UM) as a viable service offering. Although proprietary technologies and the lack of intelligence, both in the network and in access devices, have limited the development of UM to date, first generation integrated messaging services that

offer partial unification of messaging services have arrived in the marketplace. This paper investigates what is meant by a Unified Messaging System (UMS), how they are currently evolving and attempts to predict, in broad terms, what the future holds for unified messaging.

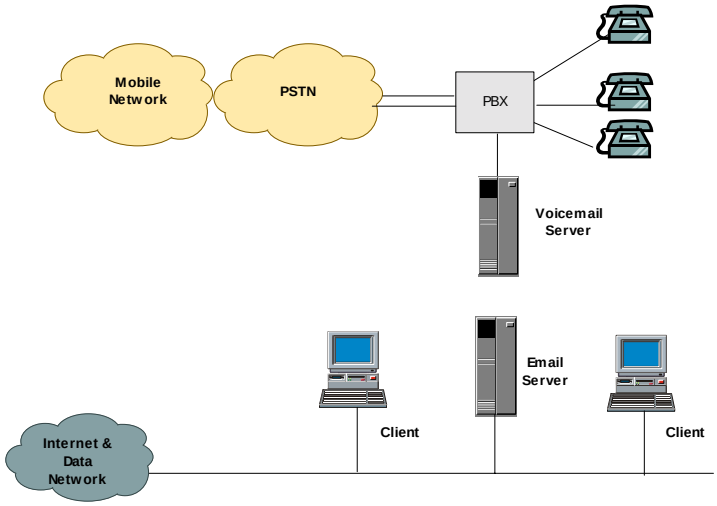


Figure 1 Typical Legacy Network Architecture in the Enterprise

2 Is there a market for Unified Messaging?

Most of the market research in Unified Messaging appears to have been focused on the US market. In recent research conducted by the Gartner Group (April 2001), it was established that nearly 50% of small businesses in the US see benefits in unified messaging and want some degree of integration among their systems. Their research is interpreted as a strong indication that " a market for unified messaging may be finally emerging". The research identified telecommuters, home offices and small businesses as the market segment most interested. Residential and private users represent less than 5% of the projected US market. The research also indicated that there is similar potential for mid-range (up to 1,000 employees) business in Canada and the US.

In a separate study, Frost & Sullivan (Press Release in June 2000) predict that the market for UM services will grow to US$5 billion by 2005. In Europe, where mobile services are well advanced and the mass market penetration is higher than in the US, it is likely that the rapidly growing number of users requiring unified messaging will increase as Mobile Service Providers add Integrated Messaging to their service bundle offerings.

3 Existing UMS Providers

UM Services can be provided in two basic ways: by the service provider (SP) or by the enterprise. If provided by the SP, the infrastructure, including the 'Unified Inbox' belongs to the SP and is located off the customer site. This approach is suited to the mass market and to smaller enterprises who are strongly reliant on service providers for their underlying communications services (e.g. mobile telephony services), who don't wish to make a high initial investment or primarily want unified messaging for contact purposes only. It has the advantages of high and rapid scalability and low initial investment costs. Instead the customer pays incremental charges for each extra 'inbox'. Service providers with existing primary services, such as ISPs or Mobile Phone operators, have been able to add some integrated messaging services quite easily by simply adding a UM platform, thereby augmenting their existing service with enhanced offerings to the consumer. A good example is the provision of integrated SMS, email and voicemail. From a technical standpoint this is a relatively simple operation and the challenge lies principally in the provision of operational service support e.g. provisioning a single account with enhanced multiple services or billing end-users at different rates depending on which service is being used.

The alternative is for the enterprise to own and operate its own UMS infrastructure, giving far greater flexibility and the opportunity to integrate UM into collaborative or other applications. This has the important advantage of improved security as a result of complete control. Given that the critical communications systems of telephony and email (and fax still, to some extent) are typically based on-site for most medium to large enterprises, it seems likely that the UMS will reside on site. In summary, it seems that for the foreseeable future, UM services for the Mass Market and SMEs is likely to be provided by external service providers and that the larger enterprises will invest in their own systems.

4 What is a Unified Messaging System?

What do we means by a unified Messaging system? A Unified Messaging System (UMS) is primarily concerned with the unification of messaging over disparate messaging systems. It is concerned with both incoming and outgoing messages and with the management of stored messages. One broad definition for a UMS might be a system for sending, receiving and managing messages, which supports multiple media types and provides access to any message from any device.

The functionality that a UMS could potentially provide includes the following capabilities:

- to send and receive messages of various types (e.g. email and sms) from within a single application
- to manage (save, delete, forward, attach, archive) all messages within a single inbox
- to provide access from a wide range of terminals/devices
- to use a common command set for access
- to use a single message store
- to convert messages from one media type to another
- to process messages automatically
- to provide Interactive Voice Response (IVR), keyboard or graphic user interfaces
- to support real-time messaging (e.g. voice calls, video conferencing) as well as non-rela-time (e.g. voicemail, email, message board) and near real-time (paging, SMS, MMS)
- to work with messages of disparate media: text, audio, video
- to create multimedia messages
- to provide a 'one number service' for a user that finds the user wherever the user is
- to provide 'personalised assistant' services
- to provide unified directory services

From this functionality we can identify the key components of the future UMS:

- Single or Unified Inbox
- Unified Command Set for Access
- Unified Message Store
- Interface to private networks
- Interface to public telecommunications networks
- Unified Directory Services
- Unified Management of Disparate Messaging Components
- Rules-based Forwarding of Messages

5 Evolving UMS Architectures – An Overview

A number of different approaches to the unification of messaging have evolved. These meet the functionality and features identified above to a greater or lesser extent. This can be categorised as:

- Integrated Messaging Systems
- Unified Messaging Systems for non real-time messaging
- Unified Messaging Systems with real-time call capabilities
- Unified Communications

Although listed roughly in the order in which they have emerged, these different approaches have not evolved from a common origin but rather from different origins based primarily on whether development began from either a telephony or email start-point. The approaches also use a different architectural model and it is worth examining these in order to understand the degree to which any given approach can meet the features and functionality of a true UMS.

5.1 Integrated Messaging (Client Focused)

Integrated Messaging (Client Focused) is the unification of similar message-type services (usually non real-time or near real-time) for access or retrieval where the individual messaging processes are kept separate. An example is the unification of email, SMS or voicemail into a single service.

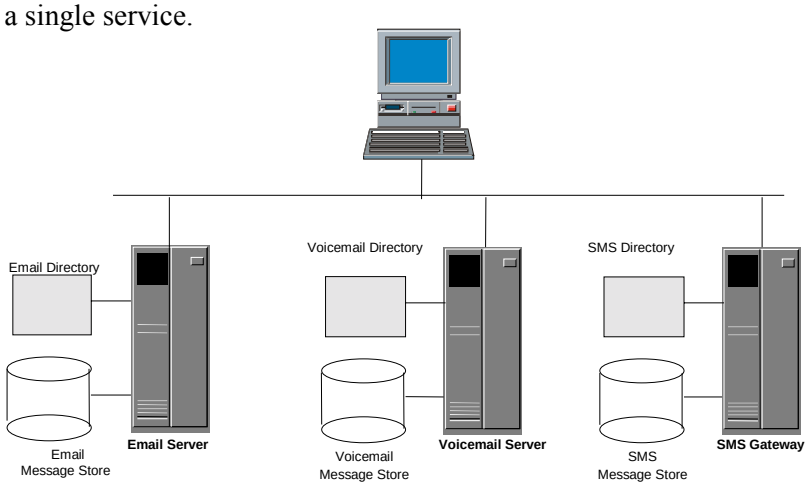


Figure 2 A Client-based Architecture for Integrated Messaging

In integrated messaging, each message service maintains its own message store, directory and administration interface. In order to unify messaging, integration software is needed to manage message traffic between multiple systems. This type of architecture pushes the core of the unification process out to the edge of the network, ultimately to the end user station. For this reason, integrated messagsing systems can be considered to be 'client focused'. Some synchronisation may also be needed between the distinct message services/servers.

5.2 Unified Messaging

5.2.1 Unified Messaging (Server Focused)

Unified Messaging (Server Focused) is also a client/server architecture and remains focussed on non-real time messaging. The main difference is that it consolidates different message type services to a single mailbox. All messages can then be accessed from within a single environment (most commonly like an email environment) and uses a single message store, a single corporate directory and a single interface for user administration. This approach typically leverages the existing email paradigm and infrastructure.

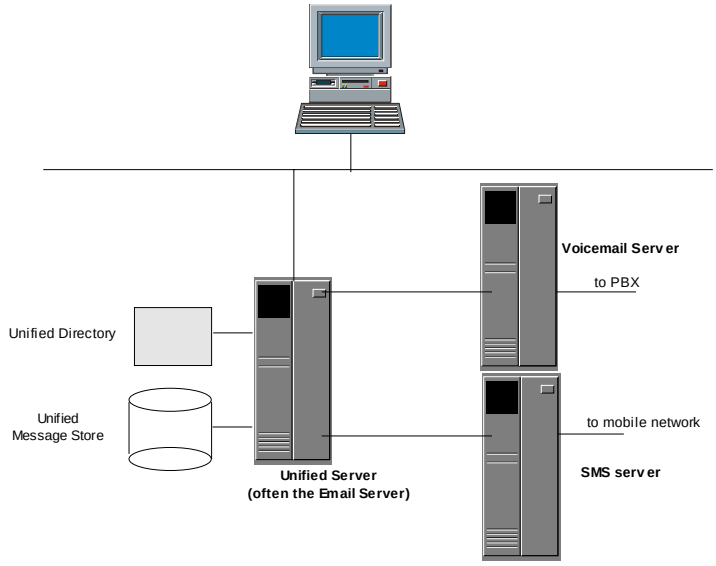


Figure 3. A Centralised Server Architecture for Integrated Messaging

It places the core of the unification process more firmly on the private network server than on the end station. Although it has the disadvantages of a single point of failure for all messaging and the need to stream traffic twice across the LAN, the server-based approach offers improved efficiencies in network administration and ease of use and is generally considered a more elegant approach than integrated messaging.

5.2.2 Unified Messaging (PBX / Switched focused) with Computer Telephony Integration

Whereas the Unified Messaging approach tends to be focused on integrating voice into an email or similar type of environment, there have also been efforts to integrate email into the PBX and switch environments.

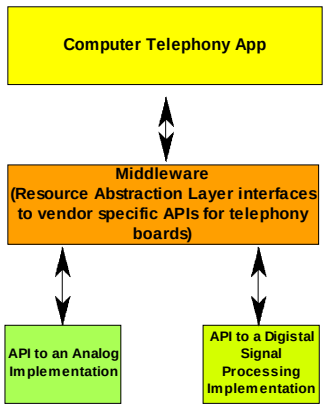


Figure 4 Computer Telephony Integration Architecture

This benefits from using the richer feature set of digital protocols that are native to the PBX rather than the integration of rudimentary analog media between the PBX and the unified 'email-type' erver. Such implementations can not only integrate text messaging services and use text-to-speech to convert emails to audio but also can integrate Computer Telephony applications into the unified messaging, thereby supporting both real-time as well as non real-time messaging. The UM applications depend on the PBX to perform switching, call routing and the transfer of message-related information. The disadvantage is that although voice systems are largely standards based, their implementations are often proprietary in nature and lack the benefits of a truly open architecture.

5.3 Partially Converged Services

With the addition of voice capabilities to an existing IP network using voice over IP technology, it becomes possible to combine real-time calls and conferencing with asynchronous messaging over an IP infrastructure. Instead of relying on a central PBX, VoIP uses the corporate LAN to switch and route calls. Applications can then be added using Application servers that use the corporate LAN. This change in the configuration greatly impacts the foundation of the technology needed to build true UM applications that supports real-time as well as non-real-time messaging – the emphasis now shifts from integration with the PBX to integration with an existing open set of LAN protocols.

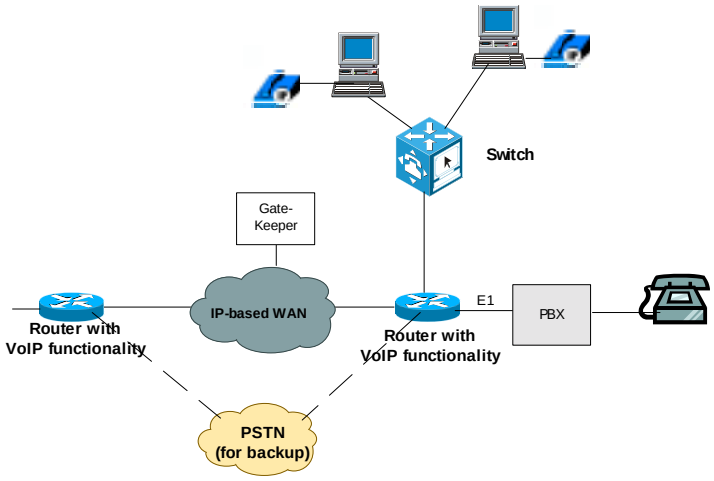


Figure 5: A Partially Converged Services Architecture for Integrated Messaging

Migration to the IP converged technology requires the use of VoIP gateways (internal or external to the PBX) in the architecture to interface existing capabilities to a VoIP infrastructure. One possible architecture is to a centralised gateway to break out onto the IP network over, say, an E1 interface. An alternative model is to use to replace the E1 interface card with a VoIP interface in the host and also use a host-based call control protocol stack (H.323, SIP, Megaco, etc.) for call establishment and tear down. DSP is still required for playback, etc. This is generally preferable as it allows a single protocol stack to control multiple gateway cards and easily facilitates protocol changes. Both approaches, however, suffer from scalability problems.

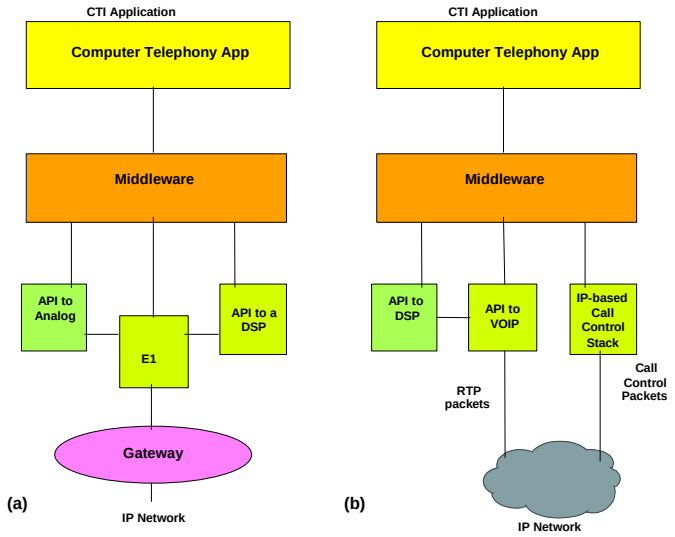


Figure 6: Migration to VoIP Technologies using (a) Centralised gateway or (b) VoIP with separate IP-based call management

An improved approach is to replace *both* the E1 and the Digital interface with an IP-based media processing board that both provides the DSP capability to record, playback, conference, etc. and also provides an IP interface.interfaces with separate IP-based call control. Although more cost-effective, it requires added complexity in the middleware and this does not improve the basic scalability problem.

5.4 Next Generation - Unified Communications

This is a converged IP network that offers voice, data, video and multimedia applications in an integrated enterprise infrastructure which uses both circuit switched and TCP/IP technology and protocols. Unified Communications are likely to operate within the enterprise over an entirely IP network because IP provides better control of bandwidth than traditional circuit-switched networks for voice and video. An application platform is still needed. It has a directory to manage personal profile information and that acts as a bridge between the circuit switched and packet switched infrastructure. Other convergent service applications can then be easily added to this basic structure. With the directory in place, advanced functions such as the use of a 'single number/find-me' are possible.

An important component of the UMS in a fully converged environment will be the insertion of intelligence into the call or message path using agent technology. This will be a key identifying characteristic of the UMS of the future. We can identify two types of agent, a Personal Agent and a Network Agent, which will be used to provide end-station and network intelligence respectively. A personal agent will operate on a unified inbox and manage all incoming and outgoing calls calls, messages and other data. It will know about the user's profile, manage the user's interface to the network, provide the user with access to network services and content through different terminal types and perform other useful functions on the user's behalf. Network agents will perform specific functions on behalf of a network provider, acting as a management agent to monitor network resources, collect useage data for billing, provide troubleshooting functions, interface to new services from other providers may be used to maintain knowledge about available net services and content, manage access to and provide access to and control of network resources.

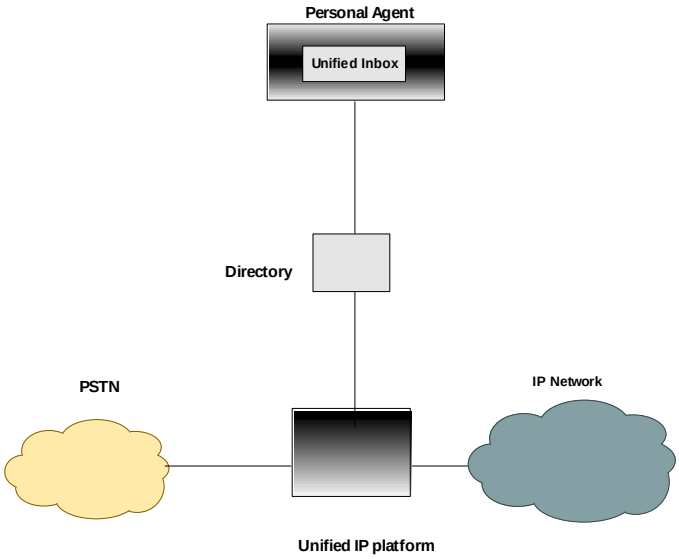


Figure 7 A Unified Communications Architecture for Integrated Messaging

Unified Communications is likely to make use of a Distributed Architecture approach which separates the media processing requirements of the application out into an external media server. The application server will retain all the logic required to execute the application and one or more media servers will be responsible for prompt intensive processing functions such as playback. The Application Server will use an advanced call and media control protocol like SIP or Megaco to control the media server. The placement of the intensive processing away from the application server will mean that far fewer CPU resources are required on the application server for each session. This architecture is highly scalable – media servers can be racked and stacked. Once again, this architecture will require a considerable effort in reworking the middleware required.

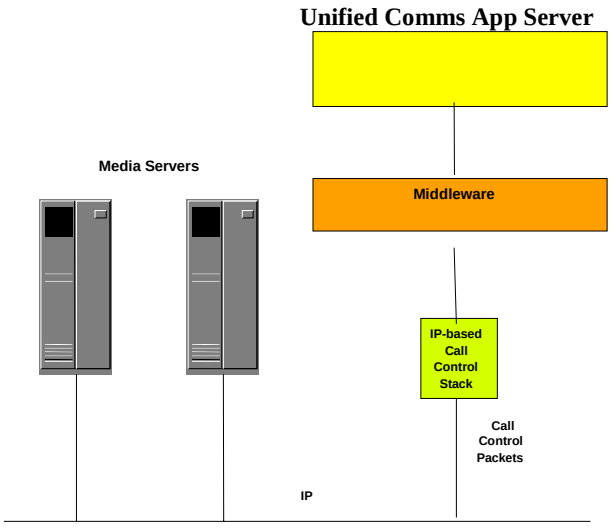


Figure 8 Telephony in a converged Unified Communications Architecture

6 Emerging Technologies that will accelerate UM

Emerging interfaces and technologies are rapidly evolving which will accelerate to UM development. Some of these are:

VoIP

Voice over Internet Protocol (VoIP) provides the capability to make a telephone call, route a call and transfer the call content over the Internet, thus providing an alternative to a traditional telephone call to anywahere in the world, but at a local toll rate. With mobile technologies such as EDGE, VoIP will also be available over mobile phones in 3G and even advanced 2.5G mobile networks.

VoiceXML

Voice Extensible Markup Language (VoiceXML) is an open standard for voice recognition and allows a user to access web based information through a voice interface. This Interactive Voice capability enables a user to navigate and retrieve non-voice related material through a voice-oriented end station, such as a phone or while on the move. It will therefore be a key access enabler.

SIP

The Session Initiation Protocol (SIP) is designed for an environment that is completely IP-based. It provides a framework for developing an enriched user experience by managing applications across a network e.g. SIP can be deployed over the internet to provide the advanced management of voice calls between any IP-compatible terminals (mobile, PDA, telephone, Workstation, etc.).

Multimedia Calls

Protocols are continuously being developed that will improve the capability to conduct real-time streaming of audio and video across the internet connection. This is particularly challenging over the limited access bandwidths of the mobile network. Multimedia Service (MMS) is providing the capability to combine multimedia in a single message.

7 Applications of UMS

The fundamental function of a UMS is clearly to manage communication using messaging from disparate systems using a single inbox. The most obvious applications arising from this might be classified as 'Contact' applications and could include the following:

Contact Applications

- Unified Contact Applications
- Telemetry Applications with Monitoring and Intelligent Alerts
- Intelligent Call Centre Applications
- Emergency/System Response Applications
- Message Filtering Applications

An implicit byproduct of UM is the use of a common directory service for all communications i.e. a unified directory service. The type of applications arising from these are:

Unified Directory Applications

- Enterprise Directory Applications
- World-wide Directory Applications

Whereas these two types of application are useful in their own rights, it is more likely that UMS will gain wider acceptance as a service that enhances other group-oriented applications which require the sharing of information and coordination of activities. Typical unified groupware applications might include:

Unified Groupware Applications

- Intelligent Information/Database Sharing applications
- Dynamic Group Diary/Scheduling applications
- Distributed Groupware Applications
- Distributed Decision-Making Applications
- Remote Learning Applications
- Virtual Office, Learning and Social Environments

8 Conclusions - The Future of UM

Evolution of Communications towards IP

The telecommunications world is making a shift away from traditional circuit-switched technologies towards a more open packet-switched infrastructure based on IP. IP based communications provide better control of bandwidth than traditional circuit-switched networks for voice and video. The traditional weaknesses of IP for real-time communication are being overcome with the emergence of advanced protocols such as MPLS which will support multimedia traffic with the varying classes of service required for mixed traffic over routed networks. This does not mean that the Internet will replace the telephone network in the foreseeable future, if ever. Nor will mobile networks become simply a way to transport IP traffic. Rather, the convergence of these technologies will be increasingly leveraged to provide enhanced services over IP.

Emergence of UM

The development of UM services has been limited by the proprietary nature of legacy voice systems, reliance on circuit witching and the lack of intelligence in both end devices and in the network. As a result of convergence towards open standards and APIs, UM is now gaining momentum. UM is not a new product but rather a set of resultant capabilities that seeks to leverage the individual benefits of the different networks in order to deliver new services which operate across all of them. Whereas the problem of integrating legacy networks based on traditional circuit-switching will remain with us for some time , the move towards VoIP in the enterprise will serve as a major catalyst in the adoption of UM.

Placement of services

Most companies provide voice calls, voicemail and email through on-site equipment. These are the critical components of unified messaging and indicate that UM services are most likely to be provided within the enterprise. The likely integration of UM with collaborative and workflow applications reinforces this view. For the mass market, the UM service will be provided through a service provider. Market forecasts indicate that the demand for unified messaging will grow significantly in the short term.

Open Platform with Components

For enterprises interested in Unified Messaging, it is becoming more important to take cognisance of convergence when making architectural choices. The convergence towards standard interfaces and open APIs is leading to the decoupling of established proprietary system hardware and software components. Core functionality previously associated with proprietary systems can now be developed as components for open platforms. Functionally optimised components can now be provided by vendors with different specialisations. This component-based strategy and the requirement for open API compatibility makes Java an appropriate choice for the development of UM services.

Approaches and Architectures

Current approaches to UM implementations tend to be based on either the Email or Voicemail paradigm. This limits the UMS to non-real-time messaging. Emerging UMS will leverage partial convergence, and in the future possibly Unified Communications, in order to provide a truly unified messaging experience for the user - an experience that includes comprehensive real and non real-time communication. Architectures are likely to be based on a set of distributed components, developed around open standard interfaces and APIs.

Applications

UM services will provide contact and directory applications and become integral to groupware and workflow applications. It will use intelligent messaging agents to help organise and route messages based on user-defined rules.

References

Douskalis, Bill. (2002). *IP Telephony - The Integration of Robust VoIP Services*

Faulkner Information Services. (2002). *Unified Messaging Market Trend*

Loshin, Peter, Paul Hoffman (2002). *Essential Email Standards: RFCs and Protocols*

Schoener, Margaret. (April 2001). *Clear Signs of Demand for unified Messaging*. Gartner Group.

"Searching ……"

Karen Church[15]

The Institute of Technology, Blanchardstown
Karen.church@itb.ie

Introduction:

The Internet is a global collection of computer networks that in collaboration provides a powerful communications service and a comprehensive hierarchy of accessible information to millions of users worldwide. The Internet is used as an information retrieval mechanism by millions of different people across the globe. Searching for information on the Internet is an everyday occurrence for some people, but partially due to the volume of information stored within the Internet's infrastructure, it is not a trivial task. On submitting a request for information, a user can be overwhelmed by thousands of results that may have some relevance but may not be valuable to the user. The process of information retrieval on the Internet is a computing subject of interest to the general public.

History:

The Internet is probably the most revolutionary concept to enter the world of computers and communications. It enables the cost-effective interaction between users and their computers, it allows easy access to the largest source of information in the world and it brings broadcasting and multimedia capabilities never before perceived to the computing industry all without any geographic considerations [1].

The first origins of the Internet date back to 1962, when J.C.R. Licklider of MIT introduced his 'Galactic Network' concept. Licklider proposed the idea of a global interconnection of computers that could be used by people to access data and programs [1]. His vision is not unlike the Internet, as it exists today. In 1961 Leonard Kleinrock of MIT published a paper on the theory of packet switching in communications, which was a huge breakthrough in the field of computer networking. Kleinrock presented his idea to MIT researcher Lawrence G. Roberts who was working for DARPA, the Defence Advanced Research Projects Agency,

[15] This paper won the Higher Education and Training Awards Council Prize for Computing 2002

researching the field of computer networking. After agreeing that packet switching seem the way forward, Roberts continued working on the project and in 1967 he published a paper detailing his plans for ARPANET, the Advanced Research Projects Agency [1].

The entire field of networking, connections, Internet protocols and communications was researched over the following years by ARPANET. From one node in 1969 to 23 by 1971, the numbers of nodes connected by ARPANET broke 1000 in 1984, 100,000 in 1989, 1,000,000 by 1992 and over 50 million by 1999. Throughout the years APRANET were responsible for a number of networking breakthroughs including the creation of the TCP/IP protocol [1]. While research and development was being carried out by scientists in the field of computer networking, a new concept called the World Wide Web was also being conquered.

Tim Berners-Lee, a CERN computer scientist, introduced the World Wide Web in 1989. The World Wide Web (WWW) is a system that essentially sits on top of the Internet and provides an information sharing mechanism. The idea behind the WWW was to provide users with a simple, single user interface written in hypertext that would enable users to browse the information on the Internet and to access that information in an easy and efficient manner [2].

The original proposal of the WWW anticipated a global hypertext system known as HTML, which had been developed by Tim Berners-Lee. This new language would enable web documents to be indexed and formatted for presentation over the Internet. A simple protocol that would process user requests for documents and present them to the user was also proposed called HTTP, the Hypertext Transport Protocol. During the course of the 1990s the concept of the WWW and the technologies behind it blossomed and the growth of the Internet and the WWW exploded to depths that were inconceivable by the mid 1990's [2].

Today the Internet and the WWW are considered a necessity for users and businesses alike. They provide companies with the ability to establish a global presence and to remain accessible to customers at all times. They enable companies to partake in eCommerce and eBusiness where business processes, marketing ventures, business transactions and customer service facilities all take place online. Users are able to interact and communicate with other people in a very cost-effective way, shop online using interactive shopping cart applications and secure payment systems, and find an abundance of information on almost any subject area.

Searching the Web:

For users to find information, products or services online, they generally employ the services of a search engine. Search engines work on behalf of a user, by receiving a query as input and carrying out a search of the Internet, in order to find relevant information for the user. Users frequently encounter problems while searching for items on the Internet for a number of reasons. The first reason is that the sheer volume of information present on the Internet makes trying to find a specific item increasingly difficult and although search engines are very useful tools for searching the Internet, they can be very inefficient. All who use the web have had experiences with search engines where we enter our keywords, click on submit and receive either hundreds of thousands of results or very few. When we do receive too many results, we are then presented with the exhaustive task of trawling through those results in order to find what we were actually looking for. Although the web and search engines provide efficient access to an increasing amount of information, there are definite limitations to current search engines and the existing method of information retrieval.

Search engines are search tools that assist users in finding items on the Internet. Search engines use software agents that survey the Web and build databases of indexed web documents that are then used to carry out searches on behalf of the user. There are a number of different types of search engines in existence but every search engine performs three basic functions [4]:

- They search the Internet for special words found in web pages.
- They store an index of these words along with the details of where to find the websites that contained the words.
- They allow users to look for words or combinations of words found in the indexes that they build.

Today search engines index hundreds of millions of pages, and respond to tens of millions of queries per day [4]. To find information on the hundreds of millions of web pages that exist, a search engine uses the services of software robots, called *spiders*. Spiders search the Internet and build lists of the words they find on web sites. This process is called *web crawling*. Most web pages have Meta tags built into them. Meta tags are keywords entered by the owner of the web page that influence the way in which a web page is indexed [4].

The data obtained by the spiders has to then be indexed and stored with additional information that makes it useful to the search engine. Normally this additional information

includes ranking and weight information that dictate the order in which web pages are returned as a result to a user query. Once the search engine has designed and built a useful index, users are then able to submit queries to these indexes in order to find the information they desire [4].

Despite the fact that search engines provide substantial assistance in locating and presenting information to users, a number of problems can occur quite frequently. The first problem is that each search works in a very different manner, covering a different domain of web sites, and requiring different syntax for queries. For example one search could require you to enclose a query in quotation marks, while another search engine might require you to use Boolean operators such as AND and OR to refine you query. It is very apparent that to carry out a keyword search in the most efficient manner, a user must have a comprehensive understanding of the mechanics behind search engines, which is something that the majority of Internet users do not have [5].

The second problem with search engines is regarding the results you receive after submitting a query. In general users either receive too many results or two few results. Most search engines order the search results using ranking information. A number of factors influence the ranking of websites. These include Meta tags, web page content, the format of your HTML, etc which although affect the ranking of a website, have nothing to do with the value of the information within the web pages. The accuracy of a users search is also influenced by the number of keywords you enter, which makes very little sense when it means that if a user enters lots of keywords to ensure precision, they are inevitably and unconsciously reducing the relevance of their search [5].

Another issue, and probably the most exasperating problem is that once you complete a search and finally find what you are looking for, you can't easily return after a number of days or weeks to the same result set, unless you remember the search engine you used and the keywords you entered or you remembered to book mark the web page [5]. Ideally every user should be able to carry out an Internet search that returns personalized, relevant and valuable results in an understandable format and to which a user can return when ever they wish.

Time for Change:

What if a system existed, that not only enabled the user to carry out searches that returned relevant and valuable results, but it did it in a manner that removed the current search effort away from the user? What if the traditional mechanism of user and business interaction was completely changed? In the traditional mechanism a businesses will build an interactive website that carries descriptive examples of the product ranges and services the business has to offer. The business will then deploy the website online, partake in some advertising and search engine submission procedures and essentially wait for customers or users to find them. Customers, on the other hand have to carry out all the searching in order to find information about a particular business. Imagine a system in which the current roles were reversed. A system that reduced the present search effort of the customer, and placed emphasis on the business carrying out the work in order to access more customers. This suggested system would involve replacing the traditional information retrieval mechanism with an improved system that allows the customer to simply state what they want, sit still and wait to be found while the business carries out the searching.

I would like to suggest a system that could be used to change the way the Internet currently operates in order to address the various information retrieval issues I outlined previously. The system would be able to reduce the tedious search process that customers have to participate in and place onus on the business to find customers using two new and exciting technologies to provide its functionality, namely Intelligent Agents and Web Services.

According to Professor Michael Wooldridge, a pioneer in intelligent agent systems, an Intelligent Agent is defined as [6]:

> *"a computer system that is capable of flexible autonomous action in order to meet its design objectives."*

Autonomous is defined in the English dictionary as independent, self-sufficient, self-ruling and self-directed. For an agent to be autonomous it means that it can operate efficiently without intervention or interruption from humans or from other agents. To be flexible, an intelligent agent must present a number of key capabilities. The capabilities listed below are adapted from three sources [6], [7], [8]:

- **Responsive** – intelligent agents should be able to *perceive* or sense their environment. The intelligent agents should also be able react in a sensible and well-timed manner to changes that occur in their environment.
- **Proactive** – intelligent agents should be able to display goal-oriented behaviour. This means that the agents should not simply respond to their environment, they should *take initiative* when and where it is appropriate.
- **Social** – an intelligent agent should be able to communicate, collaborate, and interact with other intelligent agents and humans in order to solve problems and to meet their design objectives.

Web Services have emerged as a new and exciting technology destined to revolutionise the way in which businesses operate over the Internet. Using web services technologies; businesses can publish their business processes, locate and subscribe to other web services, and exchange information in a very efficient manner.

The Stencil Group, a research and analysis firm that specialises in web services technologies define a web services as [9]:

> *"Loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols."*

The definition above specifies that the first attribute of web services as loosely coupled. More traditional applications depend on the tight interconnection of all the auxiliary elements. Tightly coupled systems require developers to thoroughly understand and control both ends of the connection between the elements. Tightly coupled systems also make it very difficult to extract one element and replace it with another element. Loosely coupled systems require a simpler level of management and enable flexible and easy reconfiguration of component elements [9].

The second attribute is that web services are reusable software components. Web services allow developers to reuse code that has been created by other developers by allowing the developers to assemble and extend the code blocks in new ways. The third attribute, to semantically encapsulate discrete functionality, means that web services are self-contained components. By providing information about the inputs and outputs they expect, other

software is able to invoke its functionality. Programmatically accessible means that web services were originally designed to operate at code level, meaning that they are not designed for direct human intervention. They enable software-to-software interaction, although they can be programmed to accept human interaction. They are distributed over the Internet and make use of existing standard Internet protocols such as HTTP. By leveraging existing Internet protocols, web services provide a standard component-based architecture [9].

In this suggested system the customer should be able to find the items they are looking for with minimal effort and businesses should be able to access more appropriate customers in a more efficient manner. The suggested system consists of three major components – a Customer Intelligent Agent (CIA), a Business Intelligent Agent (BIA) and an Agent Portal (AP), that in combination will enable a customer to make a request and receive what they actually want in a timely fashion, and will empower businesses with the tools to carrying out searching in order to find new customers. The system will allow businesses to register themselves as available and customers will be allowed to register themselves as seeking service. The power in the system will not be in the availability of customer data and supplier data but in the manner in which it can be matched and retrieved, giving best fit to both business and customer, easing the use of the web and enhancing the overall customer service.

The Customer Intelligent Agent (CIA) can be thought of as a type of personal assistant or intelligent servant that will work on behalf of the customer. The CIA presents the customer with timely and relevant search results with minimal intervention or interaction from the customer. When a customer wants to find something on the Internet he/she will input a request to a CIA. The CIA will take the request from the customer, process and structure the request in order to extract the real meaning and post the request in its new format to the Agent Portal (AP). The goal of the CIA is to quickly find what the customer actually wants.

The Business Intelligent Agent (BIA) can be thought of as the businesses own cost-effective, reliable and flexible sales person that will work on behalf of the business. When a business wants to find customers, it inputs specific information to the BIA including details on the products and services the business has to offer, the location of the business, the type of customers that the business wants to attract, etc. The BIA will then process and structure that information into a specific format so that it may use this information to find customers for the business. In order to find customers the BIA searches all customer requests that are posted in the Agent Portal (AP). The BIA scans the Agent Portal (AP) looking for customer requests it thinks it can handle. If the BIA finds a request it thinks its business can satisfy it makes contact with the CIA holding that request and the two agents begin to communicate in order

to conclude as to whether or not the business can satisfy the customer's request. If the request can be satisfied, both the customer and the business are notified.

The Agent Portal (AP) can be thought of as a multi-agent web-based system in which CIA's and BIA's communicate and collaborate in order to satisfy their goals. The AP acts like a type of bulletin board where CIA's post customer requests. BIA'S scan the AP searching for customer requests that it thinks its business can satisfy.

Both the CIA's and BIA's would have to exhibit intelligent, flexible, social and adaptable behaviour. They would be expected to make decisions on behalf of their owner's. For BIA's this means deciding whether its business could handle particular customer requests or not. For CIA's this means that when they receive notification from a BIA that a request may be satisfied, the CIA must decide whether or not to accept the offer made by the BIA. Both agents must also interact with their owners and keep their owners informed of any developments as they occur. These Intelligent Agents would be exposed as web services, allowing them to be flexible, open and interoperable. Using web services to expose the Intelligent Agents allows the system to take advantage of the open, dynamic and platform independent features of web services, as well as the aptitude and flexibility of Intelligent Agents. In order for an intelligent agent to be mobile, the agents need to share a common platform with its colleagues. The best candidate for providing a standard platform would be web services technologies.

At present, no such system exists that uses Intelligent Agent and Web Services technology to reduce the search effort from customers, to place responsible with businesses to carry out the searching and for customers to simply make their request and wait. The closest types of systems do involve the use of agent technology. They take the form of software agents that remove some of the tedious tasks associated with searching the Internet away from user. The majority of these systems simply take a search query from a user and simultaneously submit the search to multiple search engines, and then apply some filtering to the results. Although such software systems remove some of the search effort away from the user, the results are still not relevant enough. The agents are still using existing search engines to satisfy user requests and although what the user is looking for may be on the Internet, it may not be ranked high enough amongst the top search engines to even be included in the search agent's results.

Conclusion:

In my discussion of the Internet and its use to customers as an efficient information retrieval mechanism, I have posed the problems and inefficiencies that exist with the current search mechanism. I have provided an overview of search engine technology as a tool that facilitates users in finding information on the Internet and I have reviewed the problems with current search engines. I introduced a new system that could change the current operation of the Internet by reducing the current search effort made by the customer and by placing the responsibility of searching on the business. In using such a system the customer should be able to find the items they are looking for with minimal effort and businesses should be able to access more appropriate customers in a more efficient manner. The suggested system would provide its functionality by combining two exciting technologies, namely Intelligent Agents and Web Services.

References:

1 "*A Brief History of the Internet*", Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, Stephen Wolff, See: http://www.isoc.org/internet/history/brief.shtml

2 "*History of the World Wide Web*", See: http://www.wdvl.com/Internet/History/

3 "*A Helpful Guide to Web Search Engines*", See: http://www.monash.com/spidap4.html

4 "*How Internet Search Engines Work*", See: http://www.howstuffworks.com/search-engine.htm

5 "*Why search engines are clueless*", See: http://mappa.mundi.net/inform/archive/inform_0169.html

6 *Wooldridge, M. (1996),* "An introduction to multi-agent systems"

7 *Wooldridge, M. and Jennings, N. R. (1998),* "Applications of Intelligent Agents"

8 *Wooldridge, M. and Jennings, N. R. (1995),* "Intelligent Agents: Theory and Practice", *The Knowledge Engineering Review*

9 *The Stencil Group web services definition. See http://www.stencilgroup.com/ideas_scope_200106wsdefined.html*


ITB