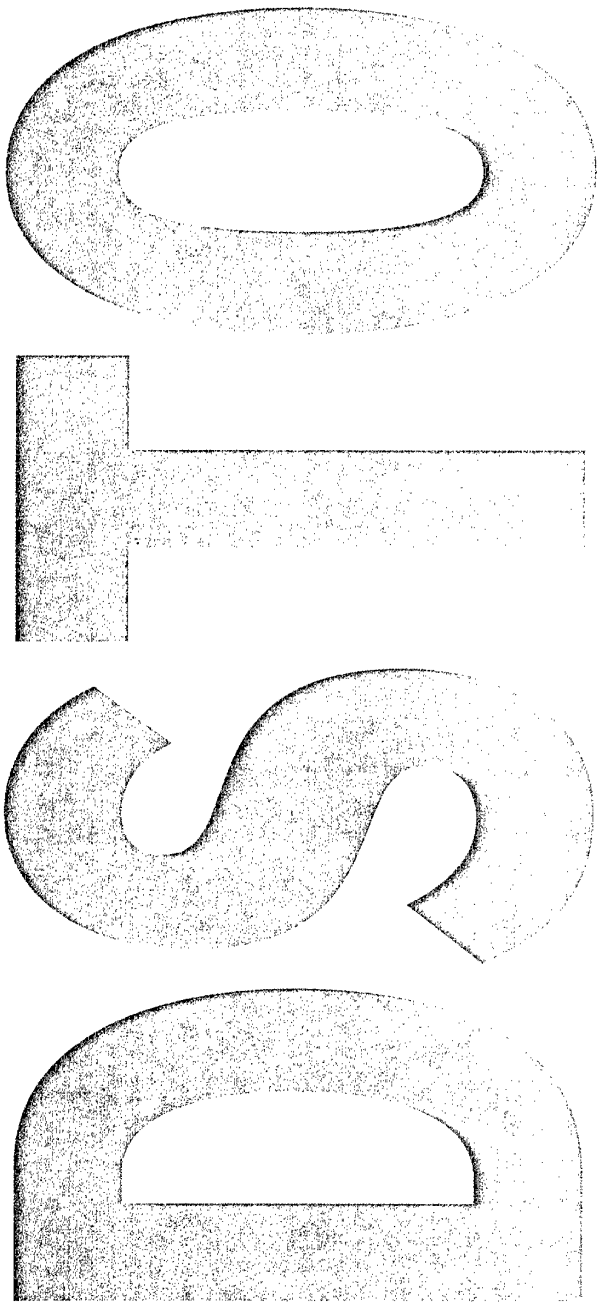
Australian Government
Department of Defence
Defence Science and
Technology Organisation



Transcription of Multiple Speakers Using Speaker Dependent Speech Recognition

Andrew Zschorn, Jason S. Littlefield, Michael Broughton, Barry Dwyer and Ahmad Hashemi-Sakhtsari

DSTO-TR-1498

Australian Government
Department of Defence
Defence Science and
Technology Organisation

Transcription of Multiple Speakers Using Speaker Dependent Speech Recognition

Andrew Zschorn, Jason S. Littlefield, Michael Broughton, Barry Dwyer and Ahmad Hashemi-Sakhtsari*

Command and Control Division
Information Sciences Laboratory

* Department of Computer Science, The University of Adelaide

DSTO-TR-1498

ABSTRACT

Speech recognition can potentially aid in producing transcripts of discussions, interviews, meetings, and other collaborative efforts. This report presents a concept demonstrator of an automatic transcription system that produces text and audio records using speaker dependent speech recognition. It is defined and discussed in terms of how it works, how users operate it, and its similarities and differences with other transcription systems.

Published by

DSTO Information Sciences Laboratory
PO Box 1500
Edinburgh South Australia 5111 Australia

Telephone: (08) 8259 5555
Fax: (08) 8259 6567

AR-012-901
September 2003

APPROVED FOR PUBLIC RELEASE

Transcription of Multiple Speakers Using Speaker Dependent Speech Recognition

Executive Summary

Computer driven transcription potentially has many advantages over other forms of meeting records.

Handwritten notes can be illegible, incomplete, and difficult to search. They can also require too much of the minute-takers' attention, thereby distracting them from participating in the meeting itself. A straight tape recording of a meeting is hard to annotate, creates too much data and is difficult to search through. The use of speech recognition has the potential to overcome these problems.

Speech-to-text transcription has become more commonplace since the availability and performance of commercial-off-the-shelf products capable of dictation have increased. Products such as *IBM*'s *ViaVoice* and *Lernout & Hauspie*'s *Dragon NaturallySpeaking* (*NS*) are two such products, which provide users of even basic personal computers access to large vocabulary continuous speech recognition. These automatic speech recognisers (ASRs) can perhaps be used to do automatic transcription.

Our goal is to develop a prototype model of such a transcriber as a concept demonstrator. This prototype is to use an off-the-shelf speech recogniser and avoid relying on specialist hardware. As well as attempting a text transcription of a collaborative session, the prototype should have the ability to make an audio recording. This audio would serve as a fail-safe record of the session, where the speech-to-text record may be incorrect due to the inadequacies of the speech recogniser.

This prototype can then be used to demonstrate the concept of an automatic meeting transcriber to various members of the Australian Defence Organisation. Possible applications of the system range from improving the interviewing process conducted in the field, brainstorming sessions, recording normal meetings or briefing sessions, and many others. The prototype seeks to improve the recording, searching and browsing of records of these events.

We have named this prototype *AuTM*, for Automatic Transcriber of Meetings. *AuTM* is based on a client/server model, with *Clients* linked to a *Server* over a normal TCP/IP (the Internet standard) network. Thus, all the computers taking part in an *AuTM Session* must be physically connected to the same network that supports TCP/IP.

The moderator (the chair or the minute taker, for example) of the *Session* runs the *Server* program. This program controls the information flow and progression of the *Session*, and the *Client* programs attach to it to join the *Session*. The moderator can use the *Server* to quickly annotate the meeting record with information such as the current agenda item, action items, or motions and their outcomes.

Each meeting attendee requires a computer and a headset microphone. On this computer, he or she runs the *AuTM Client* program to join the *Session*. This program interfaces with the *NS* program that resides on the attendee's computer to do the speech-to-text processing. As *NS* is a speaker-dependent ASR, attendees must enrol or train it to create their own speaker profiles, which they then use during the *Session*.

All computers taking part in the *Session* must have their system clocks synchronised. This is achieved by using commonly available shareware software that makes use of the Network Time Protocol. This keeps the system clocks synchronised to within a couple of milliseconds, which was considered more than adequate for the task.

As the attendees speak, *NS* segments their speech into utterances. The start of an utterance is detected when the attendee begins to speak; the end of an utterance is detected when the attendee has been silent for a certain period of time. The *Client* takes the text recognised by *NS*, and sends it to the *Server* along with the start time and end time of the utterance. As the system clocks are synchronised, the *Server* can then order each utterance it receives on the basis of its timestamp. Thus a transcript of the *Session* is generated in real-time.

The *Server* and the *Client* programs have the ability to display this evolving transcript to their users. This window displays the text of each utterance, together with the username of the attendee who said it and the time at which they started to say it. Different coloured squares are also used to identify the speakers.

Each *Client* program uses *NS*'s segmentation of speech to selectively record the attendee's microphone signal. That is, a *Client* starts recording when *NS* indicates that an utterance has begun, and stops recording when *NS* indicates the utterance has ended. Thus, the *Client* program creates one audio file per utterance. When the moderator ends the *Session*, the *Client* programs send these audio files to the *Server* program. The moderator can then use the *Server* to correct the transcript by playing back the audio files to determine recognition errors.

The moderator can choose to save the transcript and its annotations as either a HyperText Markup Language (HTML) file, or as a *Microsoft (MS) Word* file. The HTML file format includes hyper links to the audio files, so the text transcript and the audio files can be available to review via any Web browser.

AuTM can be adapted to allow the use of other TCP/IP based collaborative tools at the same time. For instance, *MS NetMeeting* can be run at the same time as AuTM, so attendees in different places can join the same *Session*. Programs like *NetMeeting* provide other tools such as the sharing of a whiteboard, document and screen, which can make distributed collaboration a more fruitful exercise.

This prototype differs significantly from other projects with similar goals. Other such projects include the *Meeting Recorder* project at the International Computer Science Institute, *Rough'n'Ready* at *BBN Technologies* and a project at Carnegie Mellon University's Interactive Systems Laboratories). The approach taken in each of these cases is to use a speaker-independent ASR, optimised for the acoustics of a meeting room and the type of language used during meetings. None of these projects depend on having a computer per user, and there is more emphasis on a 'meeting browser', a stand-alone application that enables the browsing of time-aligned text, audio, video, and other records taken from meetings.

Speaker-independent ASRs work without the speakers having to train them. The major benefit of this approach is that anyone can enter a meeting room and have the ASR recognise their voice. On the other hand, it is an extremely difficult task to make ASRs sufficiently generic for all speakers, particularly non-native English speakers.

Speaker-dependent ASRs (such as *NS*) have the advantage of improving the more a speaker uses them.

AuTM's requirement for a computer per attendee is an expensive one, compared to the other three projects. To compensate for this cost, *AuTM* makes almost exclusive use of generic, readily available hardware. That is, an attendee can use the same laptop that they use throughout the day to be part of an *AuTM Session*.

Authors

Andrew Zschorn
Command and Control Division

Andrew Zschorn graduated with a B.Sc. in the School of Mathematics and Computer Science from the Adelaide University in 2000. After a year-long placement at DSTO on the Graduate Industry-Linked Entrepreneurial Scheme he joined the Human System Integration Group as a professional officer. His areas of interest include speech-to-text transcription, speech recognition and agent-based dialogue systems.

Jason S. Littlefield
Command and Control Division

Jason Littlefield graduated with a B.Sc. in the School of Mathematics and Computer Science from the University of Adelaide in 2000. He recently joined the Human Systems Integration Group as a professional officer in 2002. His areas of interest include Automatic speech recognition, Speech user interfaces and Speaker separation.

Michael Broughton
Command and Control Division

Michael Broughton is currently a human computer interaction (HCI) specialist within Human Systems Integration group of ITD. He has a degree in Computer and Information Science from the University of South Australia. Recent research interests include: usability of speech technology, graphical user interface design, interaction devices, and evaluation within the field of HCI.

Dr. Barry Dwyer
Department of Computer Science
University of Adelaide

Barry Dwyer's professional career began as an engineer in the defence industry, mainly concerned with flight simulation. After a period as a software consultant, he joined the staff of UniSA in 1972, and, following a year as a visiting researcher at Bell Laboratories, transferred to the University of Adelaide in 1982, where he is is a Senior Lecturer in Computer Science. His current interests centre on various artificial intelligence topics.

Dr. Ahmad Hashemi-Sakhtsari
Command and Control Division

Dr Ahmad Hashemi-Sakhtsari is a research scientist in Human Systems Integration Group. His current research work is focused on application of commercial language technology to military systems and on studying human computer interaction through speech as well as manual modalities. He manages a speech and language technology research and development task in the Human Systems Integration Group.

Contents

1. Aims

The primary aim of this investigation was to develop a prototype system that automatically generates a text transcript of multi-speaker situations. Example applications of the system include meetings, interviews and brainstorming sessions. The transcript should include information such as interruptions between users and timestamps that indicate when phrases were spoken.

The system should also record the original audio of each utterance. This feature would enable the user-correction of transcript errors after the initial automatic speech-to-text process.

Further to these main goals, the prototype should aid annotating transcripts with minutes-style information, such as action items.

The system should attempt to meet these goals without relying on special hardware or software. That is, it should use an off-the-shelf automatic speech recogniser (ASR) and normal PCs or laptops.

This prototype may be used as a concept demonstrator to various interested parties within the Australian Defence Organisation.

2. Introduction

The Australian Defence Organisation conducts an enormous number of meetings, interviews, brainstorming sessions and other such collaborative efforts, records of which range in importance from limited significance to critical. Looking at ways to improve the making of these records is therefore an area of much interest.

Handwritten notes can be illegible, incomplete, and difficult to search. They can also require too much of the minute-takers' attention, thereby distracting them from participating in the meeting itself (Janin 2001). A straight tape recording of a meeting is hard to annotate, creates too much data and is also difficult to search through (Janin 2001). Moreover, should a text record be required, the tape recording will need to be transcribed, by hand or otherwise.

Speech-to-text transcription has become more commonplace since the increase in availability and performance of commercial, off-the-shelf (COTS) products capable of dictation. Products such as *IBM ViaVoice* and *Lernout & Hauspie Dragon NaturallySpeaking* (*NS*) are two such products, which provide users of basic PCs access to large vocabulary continuous speech recognition.

These ASRs can perhaps be used to do automatic transcription. By taking care of the transcription, it can free up all parties at a meeting to direct their full attention to the meeting itself. Their use also creates an opportunity to streamline annotating, storing, indexing, searching, and browsing records.

ASR tools have improved to the point where dialogue, as spoken between humans, can be transcribed accurately enough for their records to be useful (Wactlar 2000).

2.1 Infrared Prototype

This project is part of an ongoing effort to produce a mature and deployable automatic transcription system. There was an initial prototype that used hardware modules that attached to a computer's serial port. These hardware modules used infrared transceivers to synchronise the system clocks of the participating computers (Krishnamoorthy *et al.* 2000).
By using this hardware, a PC and *NS*, a participant could produce a timestamped transcript of their own speech. At the end of the session, transcripts copied from each attendee's computer were put onto a single PC using floppy disks and a program used the timestamps to work out the order in which words were spoken. Thus, we have one single transcript that records every recognised word spoken by all the participants. The new concept demonstrator is based on this prototype.

3. *AuTM* Prototype

This section of the paper will detail the configuration and implementation of the current prototype. This prototype has been named *AuTM*, for Automated Transcriber of Meetings.

3.1 Overview

This section introduces and describes the *AuTM* prototype.

3.1.1 System Configuration

Although *AuTM* is based on the infrared prototype, this was not taken to be the only possible system design. Rather, three hardware configuration possibilities were identified from the outset.

The first configuration was that of a single microphone and computer for the whole meeting. The second configuration considered was that of each attendee having his or her own microphone, with these microphones plugged into a single computer. The third configuration to be considered was identical to that of the infrared prototype. That is, each attendee would have a microphone, plugged into his or her own computer.

Of these three configurations, the first is the most attractive, in terms of deployment, due to its modest hardware needs. It is believed that attendees would also prefer this option, as it would mean they would not have to wear or speak directly into microphones. However, it would be the most difficult option to implement. This is due to the difficulties of identifying different speakers and separating their speech through a single microphone channel in the presence of cross-talk and increased noise and reverberation.

Another difficulty with the first configuration is that of using speaker dependent ASRs to recognise the voices of multiple speakers at the same time on the one computer. In the case of *NS* version 5, this cannot be done. A work-around for this problem could be to make a live recording of the audio of meetings, and doing the speech-to-text conversion offline. This would mean a delay before the minutes are produced.

The second option avoids the problems of speaker identification by having each user speak into their own microphone. However, it has the same problems as the first option in regards to using only one computer.

The third option requires more hardware than the first two. However, because the speech-to-text conversion can be performed on each meeting participant's computer, there need be no delay in producing the minutes. Meeting records would then be assembled from these separate transcripts. Of the three, this configuration was judged to be the easiest to implement.

As *AuTM* was intended to be a concept demonstrator, requiring a lot of hardware was judged to be better compromise than having a delay in producing transcripts and being difficult to implement. This is provided the hardware is generic and readily available, as is stated in the aims of the project. So the decision was made to pursue the third configuration. Therefore, this project is based on the same idea as the infrared prototype (Krishnamoorthy *et al.* 2000).

Due to its ease of implementation and ubiquity, it was decided to use direct socket connections over TCP/IP (transmission control protocol / internet protocol) between computers to remove the infrared prototype's reliance on floppy disks. This clearly means that attendees' computers would now be attached to a LAN or WAN.

Thus the system has each meeting attendee wearing a headset microphone plugged into their own PC (or laptop), with all of these PCs connected by a network that supports TCP/IP.

3.1.2 Implementation

The design decision described above highlighted the similarities between the proposed system and an Internet Relay Chat (IRC) system – something with which attendees using *AuTM* for the first time may already be familiar. Then the obvious course was to build up a complete transcript in real time, much like an IRC session, rather than to integrate separate transcripts at the conclusion of the meeting, as in the infrared prototype.

The prototype began to take shape as an enriched voice-enabled IRC system. Each attendee uses an *AuTM Client* program to log on to an *AuTM Server*, which is run by a meeting moderator. In much the same way as the infrared prototype recorded utterance text with the start and finish timestamps, *AuTM Clients* bundle this information together and send it to a *Server*. The *Server* then relays this information to all the *Clients* in a *Session*, and in this way each *Client* and the *Server* programs have an identical transcript of the session. This method has the advantage that every attendee can scroll through the transcript during a *Session* to re-visit an early part of the discussion.

3.1.3 Communication

As stated above, the communication between the *Client* and *Server* programs is based on the Internet's TCP/IP standard. The programs send unencrypted text messages through TCP/IP socket connections to communicate.

Thus, a *Server* listens on a port for *Client* connections, and every *Client* must use this port to make two TCP/IP connections. This is due to the fact that information flows both from the *Server* to the *Client* and the *Client* to the *Server*. Problems were encountered when attempting to use one connection, in that confusion arose between a *Server* and a *Client* over which was able to write to the socket. More specifically, the *Server* seemed to have lost the events that were alerting it to more data in the socket. In order to overcome these difficulties and simplify the socket communications, two

connections are used rather than one. So one of the connections is used for *Client* to *Server* messaging and the other for *Server* to *Client* messaging. The only messages permitted to flow the 'wrong way' (e.g. from a *Client* to a *Server* on a *Server* to *Client* connection) were acknowledgements, informing the sender that their previous message was received correctly.

The purposes of messages sent from the *Client* to the *Server* include:

- to contribute a new utterance to the transcript;
- to initialise usernames; and
- to send audio files.

The purposes of messages sent from the *Server* to the *Client* include

- to inform *Clients* of a new utterance contributed by another; and
- to do 'housekeeping' such as assigning colours and numbers to each *Client* and initialising and advancing the agenda items.

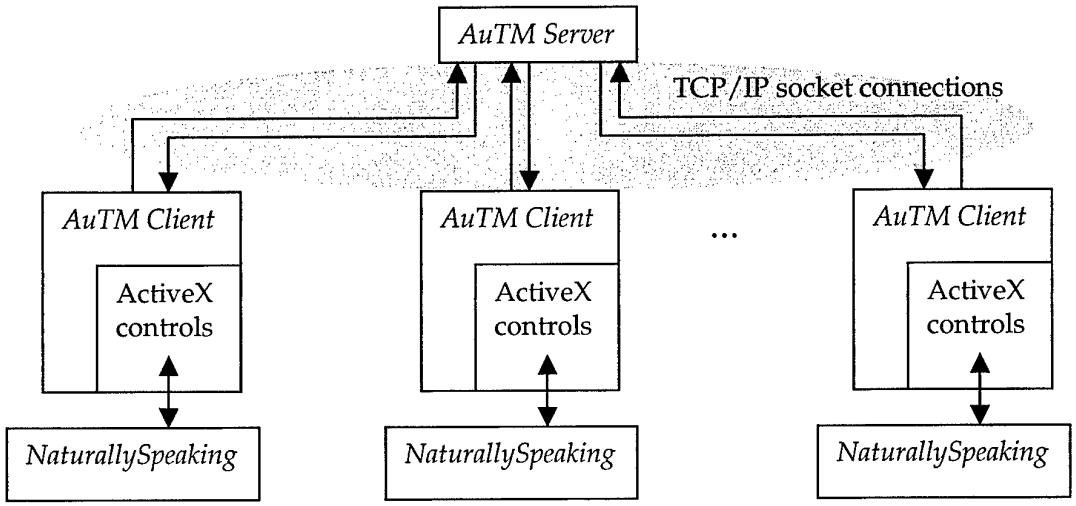


Figure 1 Diagram of the AuTM system showing each Client/NS pair executing on separate PCs, and the twin TCP/IP sockets each Client uses to connect to the Server

3.1.4 Synchronising computers over a LAN

As mentioned earlier, the major shortcoming of the infrared prototype was its reliance on the specialised timing hardware. As stated in 2.1, the role of this hardware is to synchronise the clocks of the separate computers involved in the meeting. As the *AuTM* prototype is a TCP/IP based system, it was necessary to perform this same function in a different way.

The method chosen was to employ the use of the Network Time Protocol (NTP). This protocol allows the synchronisation of system clocks over the TCP/IP. Various

Microsoft (*MS*) *Windows*-based client- and server-side shareware applications that use NTP to synchronise PC system clocks are readily available.

NTP is accurate enough to synchronise clocks to within less than a millisecond when a server and client are connected over a LAN, or a few tens of milliseconds over a WAN (Mills 1999). While this method cannot achieve extremely accurate synchronicity, the resolution is considered more than adequate for the task at hand.

3.2 Client Application

This section of the report explains the client-side *AuTM* application.

3.2.1 Role and features

To participate in an *AuTM Session*, each attendee of a meeting runs an instance of the *AuTM Client* program on his or her own computer. This application is responsible for interfacing with *NS* and with an *AuTM Server*. It displays the transcript of a *Session* and other session information to an attendee.

On executing the application, an attendee is prompted to select their *NS* user profile from the list of those available on the PC.

After the selected profile has loaded, an attendee is then asked to type in a username. This username is the familiar name used to identify the attendee to other attendees during the *Session*.

An attendee must then identify the IP number of the computer running the *AuTM Server* application. The *Client* will then connect to the *Server*, thereby commencing an *AuTM Session*.

3.2.2 Graphical User Interface

This section of the report contains screen shots and a discussion of each of the *Client*'s windows. Note that this is not a discussion on user interface design, although it includes some discussion on this topic, rather it uses the GUI to explain the nature and functions of the *Client*.

The *Client* GUI is arranged around a floating (non-maximised) *Main* window. This *Main* window displays the usernames of all *Session* participants and the *NS* microphone button. The attendee can also toggle the visibility of three other smaller windows: the *Transcript*, *Highlights*, and *Agenda* windows.

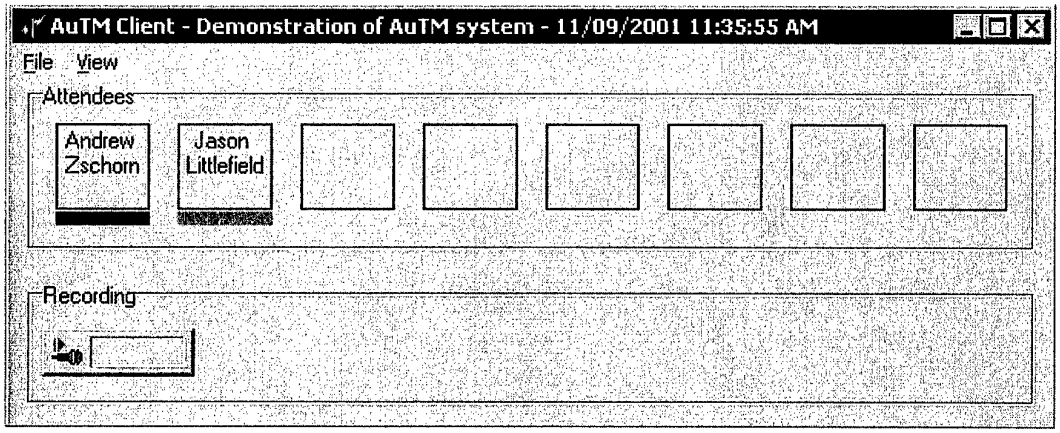


Figure 2 A screen shot of the Client Main Window during an AuTM Session between two attendees

The caption in the *Main* window's title bar is made up of the program name, the title or purpose of the meeting (as set by the moderator) and the start time and date of the *Session*. This provides basic, important information in a prominent position for the attendee.

There are two menu options. The *File* menu has an *Exit* option as its only entry. Selecting this option will cause the *Client* to disconnect from the *Server*, close its windows and halt execution. Using the *View* menu, the attendee can toggle the visibility of the *Transcript*, *Highlights* and *Agenda* windows.

The list of attendees across the top of the window allows the attendee to quickly establish the number of attendees to the meeting and their names. While there are always eight boxes for these names, they remain empty and without a coloured underline until filled by a username. This makes it very clear to the attendee the maximum number of attendees and how many there are in the current *Session*. The names of the attendees are split over two lines, such that a quick scan by the attendee across the top line is enough to be aware of their first names. Only if there is a clash of first names or if the attendees are unfamiliar with each other is it necessary to read beyond the first line. The colour underline associates each speaker with his or her identifying colour. This colour provides an additional cue to identifying a particular speaker in the *Transcript* window and in the final transcript output.

The *Recording* group box contains the user-visible components of the ASR. The *NS Microphone Button* enables an attendee to activate or mute their microphone. When activated, it displays a volume meter, which allows the attendee to determine that their microphone is set up correctly and they are speaking at a correct volume. The words that are recognised by *NS* are assembled into the *NS Results Box*, which pops-up next to the *Microphone Button*, as shown in Figure 3. The last utterance recognised remains static and visible in the *Results Box* until the attendee running the *Client* starts to speak again; it is then replaced by the next utterance.

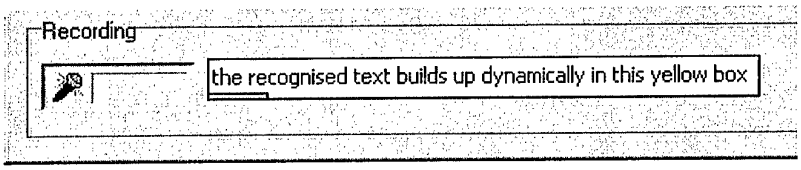


Figure 3 A screen shot of part of the Client Main Window, after NS has recognised the phrase "the recognised text builds up dynamically in this yellow box"

The attendee also has access to the *Transcript* window, via the *View* menu. The *Transcript* window, as the name suggests, displays the transcript of the session.

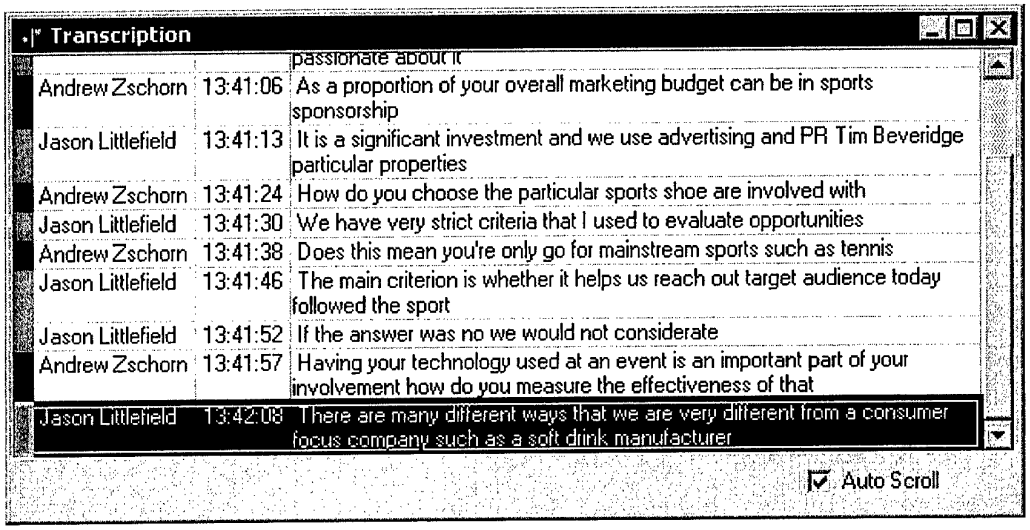


Figure 4 A screen shot of an AuTM Client's Transcript window

As can be seen from Figure 4, the *Transcript* window presents a lot of information to the attendee. The transcript is presented as a grid of text-filled cells, in which each row is an individual utterance. In the two left-most columns the attendee who contributed the utterance is identified. Both that attendee's colour and their username are used. Identifying colours are used to provides extra context for the reader (Brooks 2000). The third column displays the start time of the utterance. The fourth column contains the text of the utterance. The grid automatically scrolls up when a new utterance is added (if the *Auto Scroll* check box is ticked). When *Auto Scroll* is enabled the most recent (bottom) utterance is highlighted in reverse-video, aiding attendees' ability to pick it out amongst the rest of the text.

Figure 5 shows the *Agenda* window available to an attendee. The purpose or title of the *AuTM Session* is written across the top of the group box (in this case, "Demonstration of *AuTM* system"). The title of each item in the agenda is listed in the *group box* under the *Session* title. These items can be indented, representing a sub-section of a super-topic. In Figure 5, 'second item' is highlighted as the topic currently under discussion. As will be discussed later, the meeting moderator is responsible for inserting the contents of the meeting agenda from a text file (refer 3.3.1) and for advancing the agenda. The attendee is unable to choose which item is highlighted.

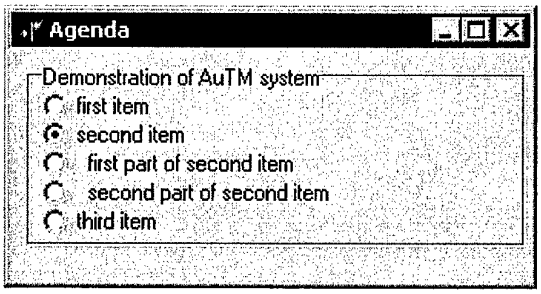


Figure 5 A screen shot of the Agenda window.

The final window visible to the attendee is the *Highlights* window, shown in Figure 6. 'Highlights' of an *AuTM Session* are those important points of a meeting record, e.g. action items and motions. As the figure shows, this window displays a dynamic list of these highlights, in much the same way as the *Transcript* window displays the utterances.

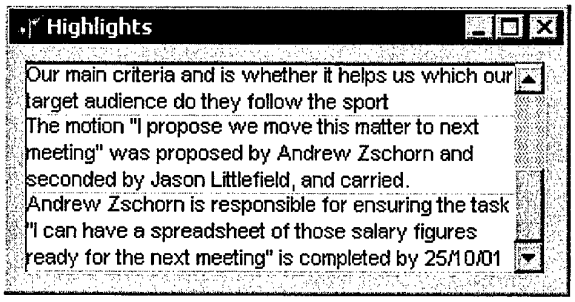


Figure 6 A screen shot of the Highlights Window

3.2.3 Speech Integration

The configuration of the *AuTM* system is such that it is the *Client* program that interfaces with the ASR. That is, the speech-to-text conversion takes place for each attendee individually, before the results are sent to the *Server*.

The actual ASR used is then independent of the *AuTM* system. This means that it is unnecessary for attendees of the same *Session* to all use the same ASR. The current *AuTM Client* was, however, developed for use with *NS*.

This section details the speech integration method used in interfacing the *NS* ASR with the *AuTM Client*.

3.2.3.1 Delphi Integrated Development Environment

The *AuTM* programs were developed using the *Borland Delphi* Integrated Development Environment (IDE), version five. This is a graphical, rapid application development IDE for *MS Windows* operating systems. *Delphi* employs an event-driven paradigm, and uses object-oriented Pascal as the programming language.

The event-driven paradigm is implemented as a pointer to methods or procedures. Objects have certain relevant events as fields (local variables). For instance, the TButton object (a basic button GUI component) has a field called OnClick, which is a pointer to a procedure. If a procedure is written and the OnClick field is set to point at it, that procedure will be executed when the button is clicked.

The method used to program in Delphi is to build up a GUI for an application, and then write (usually) small procedures that are registered as handling certain events. This means that it is unnecessary to create programs with 'main loops' and other continuously executing code. Rather, once the initial set-up code has executed, the application automatically waits for user input. User input causes events to be created, and these events trigger event handlers to execute.

3.2.3.2 *Dragon NaturallySpeaking*

Dragon NaturallySpeaking was chosen as the ASR for this prototype. *NS* is a large-vocabulary, speaker-dependent COTS ASR. It has a vocabulary size of over 250,000 words (ScanSoft 2002).

The recognition accuracy of *NS* is significantly influenced by the quality of the waveform it receives from the user's microphone. Therefore, close-talking, noise-cancelling microphones are used. These sit about two centimetres from a corner of the mouth, mounted on lightweight headsets.

As *NS* is a speaker-dependent ASR, it is necessary for every attendee to go through the training process. This allows *NS* to build up a user profile for each attendee, based on their unique voice. Reloading this user profile at a later stage will put *NS* into the correct mode to recognise that attendee's speech. Continual use and updating of user profiles will further refine them and improve the recognition accuracy of *NS*.

So for an *AuTM Session* each attendee has *NS* installed on their computer, and a trained user profile available for the speech-to-text process.

ScanSoft supplies a Software Development Kit (SDK) that enables programmatic control of *NS* (Lernout & Hauspie, 2000). The SDK includes a set of ActiveX components. These components can be directly controlled by applications developed in *Delphi*. They provide a high level of access, control and feedback from *NS*.

Like all *Delphi* components, the *NS* ActiveX controls are event-driven. Each control has a set of events, which, when fired, enable the application to maintain awareness of the state and execution of *NS*.

NS was chosen for this project because it was considered to be one of the best-performing PC-based ASRs (Glinert, 1999) and because of the ease of use and power of its SDK.

3.2.4 Speech User Interface

This section of the report explains the Speech User Interface (SUI) subsection of the *AuTM Client* program.

Broadly, there are two ways to employ *NS*. These two modes are *Command Mode* and *Dictation Mode*. *Command Mode*, as its name would suggest, is used when issuing commands to the computer. This mode is characterised by a small vocabulary as only those words that are currently valid commands need be recognised. This results in very high recognition accuracy, as there is usually a very low chance of the ASR confusing words. *Dictation Mode*, on the other hand, is used when entering prose-style text into the computer. This mode is characterised by its requirement for a much larger vocabulary, and hence much increased chance of vocabulary confusion.

The SUI of the Client is almost transparent to the attendee. The SUI was designed in this way to allow the discussion between the attendees to flow naturally and to not distract them with the speech recognition process.

In keeping with this design goal, there is no implementation of *Command Mode*. That is, the attendees cannot use any speech commands to control the *Client* program. It is believed that this will avoid the problems of: mis-recognised commands being inserted as part of the transcript; and attendees interrupting a discussion to speak commands.

Hence the *Client* implements only *Dictation Mode*. That is, *NS* attempts to recognise every word the attendee speaks, and the *Client* will forward these words onto the *Server* to be added to the transcript.

The process of the attendee contributing an utterance to the *AuTM Session* is as follows. As mentioned previously, *Client*s deal with utterances on an individual basis. When the attendee speaks, *NS* picks up the increase in signal to noise ratio (SNR) coming from their microphone. *NS* then fires an event (OnUtteranceBegin), which prompts the *Client* program to record the system time, and this is stored as the start-time of the utterance.

As the attendee continues to speak, *NS* builds up the complete string of words representing the entire utterance.

The end of the utterance is detected when the user has been silent for longer than some threshold period. If this threshold value is too small, then the gaps between words in the one sentence are incorrectly identified as the end of the utterance, and if the value is too large consecutive sentences will become a single utterance. The ideal case is when each utterance constitutes a complete sentence, which would aid the readability and ease of comprehension of the transcript. Clearly this is impossible, as people are inconsistent with the way they pause during speech. When *NS* detects this period of silence it registers the end of that utterance.

Once *NS* has finalised that text, it puts it into a text box (a GUI component that displays text) that both *NS* and the Client are able to access. In the case of the *AuTM* Client, the text box used is the size of a single pixel, that is, invisible to the user. Once the text has been installed in the text box, *NS* fires another event (OnUtteranceEnd), signalling to the Client that the utterance has ended. The Client program records the time of handling that event as the finish time of the utterance. This is only an approximation, as it is actually the time that *NS* finished recognising the utterance that is being recorded. *NS* does not recognise words as soon as they are spoken, rather it keeps a

buffer of audio that it works through at slightly less than real-time, however, on fast machines this approximation is small enough to be safely ignored.

So the Client then has the start and finish timestamps and the text of an utterance. Its OnUtteranceEnd event handler bundles this information together and sends the total to the *Server*.

The SUI thus requires very little conscious control by attendees. They simply need to speak naturally, *NS* segments their speech into utterances, and the *Client* automatically forwards those separate utterances onto the *Server*.

In casual discussion-type dialogue interruptions are quite frequent (Morgan *et al.* 2001), so this case needs to be addressed. As each attendee has their own computer, there is no problem with two or more attendees talking simultaneously as the headset microphones pick up only their owner's speech. *NS* will individually process the speech, and their *Clients* will then send the message to the *Server*, which will handle them one at a time, and will queue other messages until it can deal with them. So interruptions do not require special handling in order not to miss any text. However, interruptions are handled differently in terms of representing the utterances in the transcript. An *Interruption* utterance is identified by its two timestamps. If the start-time of an utterance precedes the end-time of the utterance before it, then it is considered to be an interruption. To highlight this, the timestamp of the interrupting (second) utterance is written in red in the transcript window.

As discussed earlier, the timestamps recorded by *Clients* in their event-handlers are approximations only. Particularly on slower computers, the end-time can be much delayed from when the attendee actually stopped speaking. For these reasons, an empirically derived threshold of 100 milliseconds is used to determine interruptions. That is, an utterance's timestamp must be out of order by more than 100 milliseconds before it is considered an interruption.

3.2.5 Recording Utterances

An important feature of the *AuTM* prototype is saving the audio from each spoken utterance made by an attendee. The purpose of saving the audio is to provide a means for manually correcting errors made by the ASR. An *Utterance Recorder* component was developed to manage the recording of the audio.

The *Utterance Recorder* has two primary features:

- to allow the speaker to select which of the computer's audio devices will be used to record utterances, and to test it; and
- to control the recording of spoken utterances.

The audio from each utterance is required in two instances: one for *NS* to transcribe speech into text and one for the *Utterance Recorder*. As mentioned earlier, *NS* is a speaker-dependent ASR, and speaker-dependent ASRs require a training process for each speaker prior to their use. As part of the training process for *NS*, the program initialises a speech engine by creating an engine-audio pair. The engine-audio pair

consists of a speech-engine mode and an audio source object (Kotmel 2000). The audio source object transfers speech data from a source, such as a microphone plugged into the computer's sound card, to the *NS* speech engine. However, the *NS* Delphi Active X controls do not allow sharing of the audio source object in its engine-audio pair. This presents the problem of how to provide the audio from each spoken utterance to *NS* and the *Utterance Recorder* simultaneously. The solution involved using a 2-way stereo audio splitter, which provides two stereo tip-ring-sleeve (TRS) audio plugs from the one stereo TRS socket. Figure 7 shows a diagram of the audio splitter.

Figure 7 Illustration of audio splitter

Hence, the configuration of the audio components consists of an analogue microphone connected to a 2-way audio splitter, which is in turn connected to two audio devices of the computer. The audio devices can include two sound cards or two USB adapters, not one of each. Figure 8 illustrates the configuration of the audio components with two Andrea USB adapters.

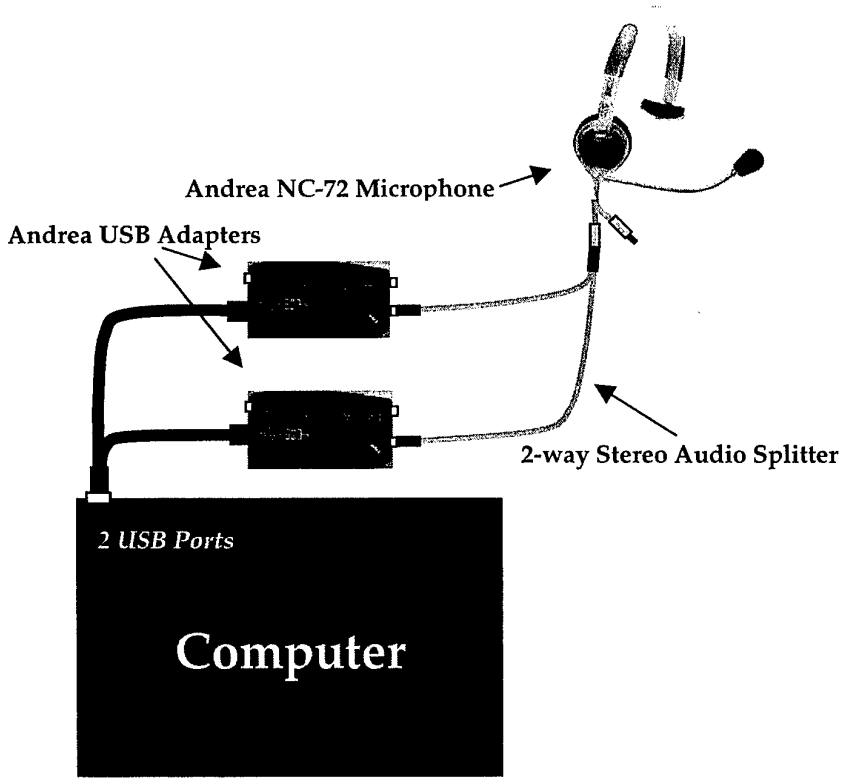


Figure 8 Configuration of the audio components for AuTM

The USB adapters provide analogue to digital conversion of the speech data from an analogue microphone for plug and play USB connectivity without the need for a sound card (Andrea Electronics 2002). Trying to use only one USB adapter and a sound card damaged a microphone beyond repair. The cause could be attributed to the Andrea USB adapter providing the bias voltage for the microphone on the tip of the TRS socket, while the sound cards provide the bias voltage on the ring of the TRS socket.

The 2-way splitting of the analogue audio signal from the microphone reduces the signal voltage of the speech data. However, preliminary testing indicated that this did not significantly degrade the performance of *NS*. If this was not the case, a more sophisticated 2-way splitter would be required involving the pre-amplification of the audio signal.

As there were at least two audio source objects, a method for selecting which audio source object would be used for *NS* and which would be used for the *Utterance Recorder* was required. The *NS Audio set up wizard* dialogue was used to prompt the user to select which of the available audio source objects are to be used for *NS*. Figure 9 below displays the first window from the *NS Audio Set up wizard*.

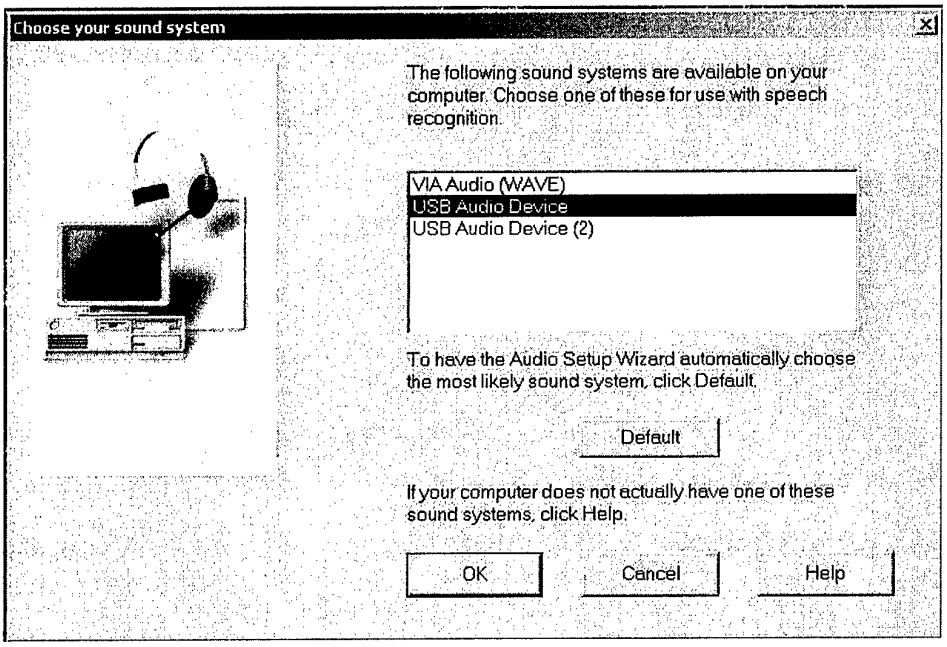


Figure 9 The Choose your sound system window of Dragon NaturallySpeaking version 5

A simple *Audio Device Options* dialogue was created to make the user select the audio source object for the *Utterance Recorder* (see Figure 10 for an example) without having to bring up the *Audio Set Up* form.

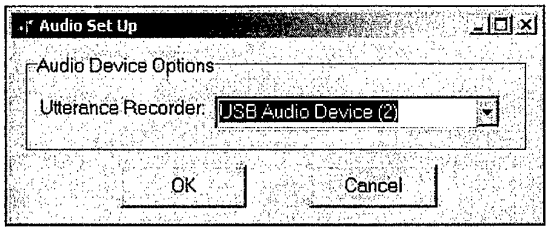


Figure 10 The Audio Device Options dialogue window

An *AuTM Audio Set Up* form provides a user interface for troubleshooting and testing audio source objects for the *Utterance Recorder*. The form is divided up into six sections:

1. The *Utterance recorder device options* selection box lists the audio source objects available and provides a means for changing the one selected for the *Utterance Recorder*.
2. The *Speech recognition device* selection box displays the audio device used by *NS*.
3. There is one *Test function* available, the *Query* function. The *Query* function displays the name and features of each audio source object found.
4. A *Status* bar displays the operational state, which include Idle, Buffering and Recording. The Status bar is sensitive to the level of the signal being recorded and is colour coded. The status bar is black when Idle, green when Buffering and red when Recording.
5. The *Query display* provides a space to display the names and features of the audio devices found following selection of the query function.
6. The *Utterance tracker* displays the name and path of the utterance audio files produced by the utterance recorder.

Figure 11 shows the *AuTM Audio Set Up* form. Note the *Query* function has been selected and the filenames of two utterances are displayed.

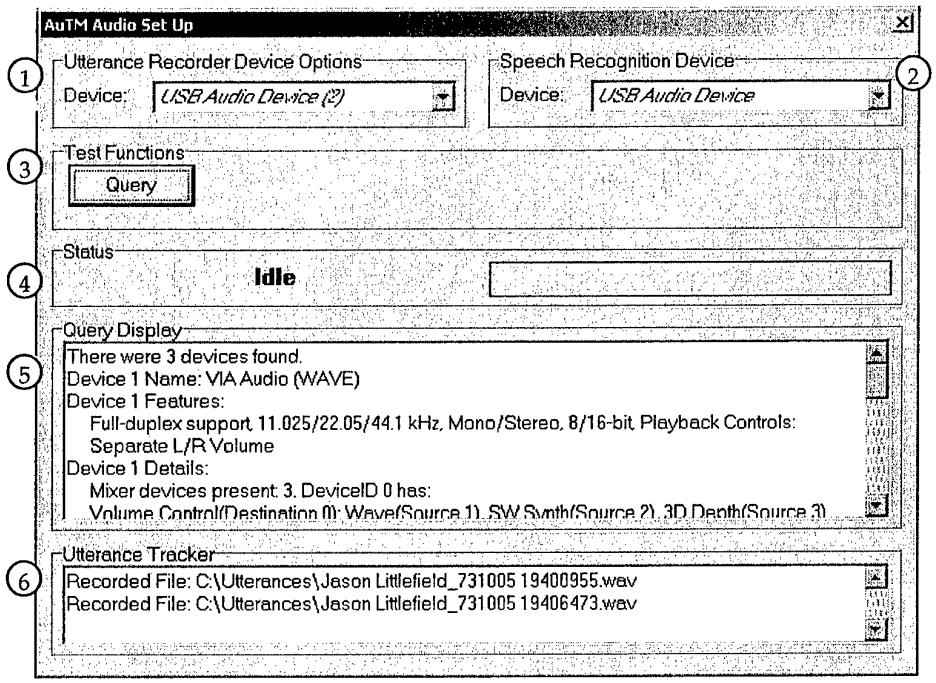


Figure 11 the AuTM Audio Set Up form

The *NS* SDK provides six Active X controls for Delphi, one of which is the *Engine control object*, which provides control of the topics, speech engine and languages. There are two events that can be triggered using the *Engine control object*: *OnUtteranceBegin* and *OnUtteranceEnd*. The *OnUtteranceBegin* event is triggered when the speech engine has detected the beginning of an utterance. Initially the *Utterance Recorder* was designed to start recording when an *OnUtteranceBegin* event was triggered and stop recording when an *OnUtteranceEnd* event was triggered. However, there was a delay between the beginning of the utterance and the event being called. This delay caused cropping of the first half second of an utterance.

The solution to this cropping problem involved using a circular buffer that utilises both the left and right channels of a stereo microphone. The buffering is activated and deactivated when the *Microphone button* on the *Client Main* window is turned on and off. When the buffering is activated, the audio signals from both the left and right channels of the microphone are recorded simultaneously. On alternate intervals of half a second the audio recording from either the left or right channel is cleared. When an *OnUtteranceBegin* event occurs, the buffer that is longest at the time continues recording, rather than being cleared. Thus the *OnUtteranceBegin* event is now a flag to continue recording (keeping about half a second of audio recorded previous to receiving the event), rather than to start recording. When an *OnUtteranceEnd* event occurs, the audio recording from the spoken utterance is saved as a mono, 16 bit, 11kHz WAVE file. Then the cyclic buffering resumes.

At the conclusion of a *Session*, *Clients* receive a signal from the *Server*, at which point the *Clients* send their audio data to the *Server*. The *Server* then re-assembles the audio files, and the moderator can listen to the audio of each utterance separately.

3.3 Server Application

This section of the report details the server-side of the *AuTM* prototype.

3.3.1 Role and features

There is one instance of the *AuTM Server* program run per *Session*, and it is the *Session* moderator who runs and controls it. The *Server* program can run on the same computer as one of the *Clients*, if the moderator is also a speaker. The purpose of the *Server* program is to centralise and control the communication between the *Client* programs and also manage the *Session* itself.

On starting the *Server*, the moderator is requested to identify an agenda file, which is used to read in the *Session* agenda. This file is a simple text file, in the following format: the first line contains the title or purpose of the *Session*, the next line is blank and each line after that to the end of the file contains the title of one item. Tab characters are used to indicate the nesting of agenda items, with a sub-item having one more tab character at the start of its line than its superior. If the moderator chooses to identify a correctly formatted file at this point the Server will parse the file and then display a representation of the agenda in the *Server's Agenda* window. This agenda cannot change during the *Session*.

After loading the agenda, the *Server* then opens up its TCP/IP port, waiting for connections from *Clients*. *Clients* must achieve two successful connections to the *Server* before they are considered part of the *Session*.

In order that the *Server* correctly pairs these separate links, it stores the IP and port numbers from which each first link is made. When a second link is made, the *Client* re-sends this information so the *Server* can find the existing connection to pair it with. This means that multiple *Clients* can connect from one machine, and that *Clients* do not have to make their pairs of links one at a time. (The *Server* can tell whether a new connection is a *Client's* first or second link by analysing the first message that it receives through it.)

At this stage *Clients* also provide their *AuTM Session* username, as chosen by their attendee. If the given username clashes with another username already in use in the *Session*, the *Server* will send an error message to the Client, and that attendee will be requested to enter a different username. Even in the case of a username clash a *Client* will succeed in establishing a first connection. This is because the *Server* identifies *Clients* by their IP and port numbers rather than by their usernames.

Clients then attempt to make a second link. The port a *Client* uses will be different to the first connection, as they are quite separate and independent. The first message a *Client* sends on making the second connection is the IP and port it is using for its first

connection. This is so the *Server* can identify which of its active connections is the first link that *Client* made. If the *Server* cannot find a first connection it will reject a *Client's* attempt to connect a second time. If the *Server* can find the first connection, then it goes ahead with establishing the second.

As the second connection is used for the server-to-client messages, at this point the *Server* can begin to send control messages to the *Client*. The first of these messages informs *Clients* of their client number and colour. Client numbers are used to identify clients in the messages, to avoid relying on usernames which are strings and therefore inefficient and awkward to use. Client colours are used to aid identification of attendees in the transcript. At this point, the *Server* also informs every *Client* of all the usernames, client numbers and colours of other *Clients* that are already part of the *Session*. The *Server* also sends the meeting title and agenda items to *Clients*. If a *Client* joins a *Session* after transcription has already begun, then the *Server* will bring it up to speed by sending it the entire current state of the transcript. This concludes the processes undertaken when an *AuTM Client* connects to the *AuTM Server*.

After the intensive set-up stage, the role of the *Server* is much reduced. Once all *Clients* have connected and housekeeping messages are out of the way, the *Server's* only significant purpose is to echo the utterance messages it receives from one *Client* to all others. The process of a *Client* building utterance messages was described in 3.2.4. When the *Server* receives such a message, it first splits the message into its parts. This is so the *Server* can build up its own transcript of the *Session*, and because the message that a *Client* sends to a *Server* is of a different format to one that is sent from a *Server* to *Clients*. That is, the *Server* must unpack, store the parts of and then reassemble in a different format, each of the utterance messages. So, when the *Server* receives an utterance message, it first commits that information to its own record of the *Session*. It then re-packages the information to copy out to all *Clients*. Note that it sends the message to all *Clients*, including the one that sent the original utterance message. When *Clients* receive the utterance message from the *Server*, they all update their displays accordingly.

The moderator can use the *Server* program to perform various other tasks in creating the meeting record. This includes the ability to compose and send a short text message to the attendees. This message is separate from the transcript, and thus is a way for the moderator to communicate with the attendees 'off-the-record', for example, when setting up the *Session*. The moderator can also select the current agenda item, which the *Server* then relays to *Clients* so they can update their *Agenda* windows for the attendees.

AuTM incorporates the concept of Highlights. Highlights are a way of recording the most important parts of a *Session* as English sentences. Along with the moderator being able to simply type in the text of a Highlight, the *Server* also provides functionality to increase the speed of entry of common Highlights, such as action items and motions. This is discussed in more depth in 3.3.2. Once the moderator has entered a highlight, the *Server* sends it as a message to the *Clients*. The *Clients* then update their *Highlight* windows so the attendees can see the changes. This means that each attendee

has a complete list of the *Session* Highlights on their desktops, which they can scroll through.

The moderator can end the *Session*, whereupon the *Server* finalizes the transcript and requests the *Clients* to upload their audio files. As the files are currently stored as uncompressed *Windows* PCM 11,025Hz 16-bit mono wave files, this takes some time. Once this process has finished, the socket connections with the *Clients* are all closed.

The *Server* then scans the transcript to tidy it up. If an attendee contributed two or more consecutive utterances, then a new utterance is created with the start time of the first utterance of the series, the end time of the last utterance and the text of all the utterances in the series. The text of each utterance in the series is concatenated, with the first letter being made a capital, and appending to it a full stop and two spaces. That is, each utterance in the series becomes a single sentence, and the series becomes a paragraph. This is done in order to improve the simplicity and readability of the transcript.

At the same time as the text of consecutive utterances are being joined, the same thing happens to the audio files of those utterances. As the *Server* has one audio file for each utterance of every user, when the texts of some utterances are joined, the corresponding audio files should also be joined.

The moderator can double-click on any row of the *Transcript* window to edit that utterance. This is explained in more detail in 3.3.2.

When the moderator has finished editing the transcript to their satisfaction, they can save it. There are two formats in which it can be saved: as an HTML file (in which case one can use any web browser to view it), or as a *Microsoft Word* document.

The HTML transcript is created by including HTML formatting tags around the various sections of the transcript, such as: the title of the *Session*, the names of attendees present, the Highlights, the agenda and the transcript itself. There is a sample HTML transcript in Appendix A.

The transcript file includes a table that is similar to the *Transcript* window. The file's table has two extra columns, one for the agenda item and another for the utterance end-time. The utterances are time-aligned with the agenda items, to indicate which utterances were spoken during each agenda item. The end-time is included to provide more information about the transcript. This extra information introduces the ability to represent two types of interruptions in the transcript. The 'overlap' interruptions are those in which an utterance's start-time precedes the end-time of the utterance before it. In this case, the interrupting utterance's start-time is highlighted, the same as in the *Transcript* window. The 'within' interruptions have their start-time and end-time precede the end-time of the utterance before it. In this case, both timestamps of the out-of-order utterance are highlighted.

The table also includes a hyperlink to each utterance's audio file.

The *Word* document is created by opening up a temporary copy of the HTML version in *Word*, and then saving it as a *Word* file (a file with the ".doc" extension). Note that this process is carried out programmatically, i.e. no user invention is required.

3.3.2 Graphical User Interface

The design and appearance of the *Server's* GUI is very similar to that of the *Client*. It is also based around a floating *Main* window, and has the same child windows: the *Transcript*, *Agenda* and *Highlights* windows. However, there is more functionality available through the *Server* versions of these windows.

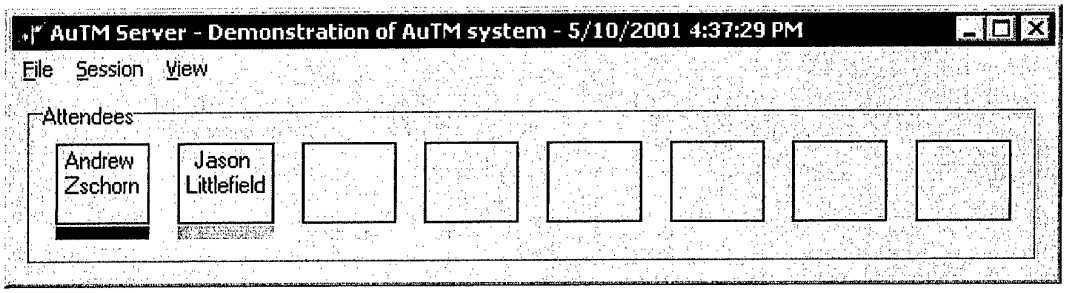


Figure 12 A screen shot of the Server Main window

As Figure 12 shows, the *Server Main* window is similar to the *Client Main* window. They both have the eight boxes of two-line attendee names. The differences are that the *Server Main* window lacks the *Recording* group box (where the *NS Microphone Button* and *Results Box* are situated), and it includes the *Session* menu. As the *Server* is designed to be entirely keyboard and mouse driven rather than voice enabled, there is no need for the *Recording* group box. The *Session* menu includes one item, the *End Session* command. The moderator selects this item when he or she considers the meeting to have ended, and this triggers the concatenation process discussed in 3.2.1.

As with the *Client* program, the *Server* has a *View* menu, through which the moderator can toggle the visibility of the *Transcript*, *Agenda* and *Highlights* windows. These windows all appear identical to the *Client* versions (shown in Figure 4, Figure 5 and Figure 6). The *Highlights* windows are also identical in function, while the other two windows have more functionality as *Server* windows than they do when part of the *Client* program.

The moderator can select the item currently under discussion in the *Agenda* window, whereas the attendees' *Agenda* windows are 'read-only'. As the moderator clicks on each agenda item every attendee's *Agenda* window is updated accordingly. The time that the moderator clicks on an agenda item is recorded, so that when creating the document versions of the transcript, the agenda items' timestamps can be used to line up the agenda flow with the utterance records.

The moderator can add Highlights (see 3.3.1) to the meeting record by right-clicking on an utterance in their transcript window. This causes a small menu to pop-up, giving the moderator the choice of using that utterance as the basis for a new motion, action

item or ordinary text Highlight. Depending on which of the three choices, one of the three following windows will then be displayed.

Figure 13 A screen shot of the New Highlight window

This is the window that is displayed when the moderator selects the ordinary text Highlight option. The recognised text from the utterance is inserted into the text box, where the moderator can edit the text to make it suitable to enter into the meeting record. This happens when the OK button is clicked.

Figure 14 shows the window that is displayed when the moderator selects the 'action item' option. The moderator first picks out the attendee (or attendees) who is responsible for the action item from the list of all Session attendees. The recognised text from the utterance is inserted into the Description text box, where the moderator can edit it. The moderator can then use the date picker to select the 'due date' of the action item, and then commit it to the meeting record by clicking the OK button.

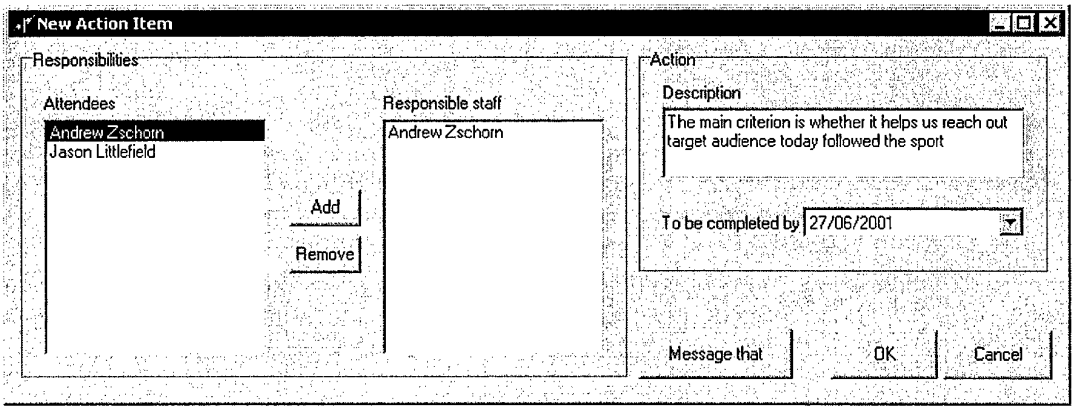


Figure 14 A screen shot of the New Action Item window

Figure 15 shows the Motion window. The recognised text from the utterance is inserted into the text box, where the moderator can edit it. The proposer and seconder

are then chosen from the combo boxes, each of which contain a list of attendees. The moderator can then choose the motion's status. The Highlight is committed to the meeting record when the moderator clicks the OK button.

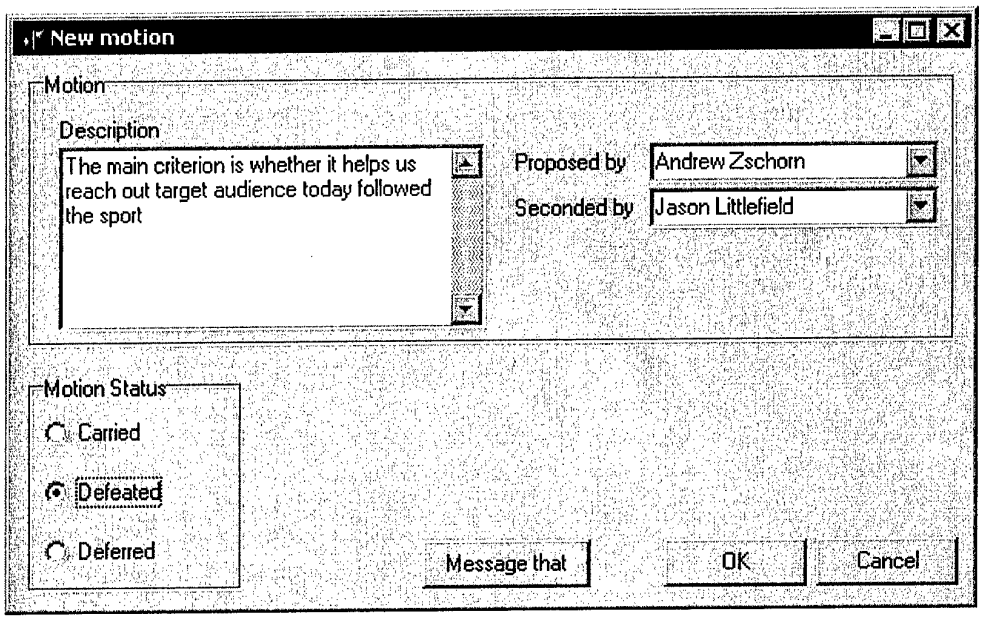


Figure 15 A screen shot of the Motion window

When any of the Highlights are committed to the record, the Highlight windows of the moderator and each client are updated.

The purpose of these mechanisms is to increase the speed at which the moderator can enter this information. By selecting a particular utterance to base a new highlight on, the moderator avoids having to type each item 'from scratch'. The *Action Item* and *Motion* windows are timesaving ways of entering an ordinary text Highlight. Once the moderator has used the GUI components to quickly enter the important information, the *Server* takes each bit of information and forms a proper English sentence.

Each of the *Highlight* windows has a *Message That* button. Clicking this button causes the *Server* to send a message to each *Client*, containing the text of the Highlight. These messages pop up in front of *Client* windows, so it is brought to the attendees' attention. This function could be useful when voting. The moderator could enter a new motion and then click the *Message That* button so that each attendee can read the motion as it will be entered into the records before they vote on it. Note that the Highlight is not entered into the record when the moderator hits the *Message That* button.

Once the moderator has ended the *AuTM Session* and the utterances have been concatenated, they can edit the utterance records. By double-clicking on an utterance in the *Transcript* window, the moderator can bring up an *Utterance* window (Figure 16).

From this window the moderator can play the original audio of the utterance, and use it to identify and correct any recognition errors. When they click the 'OK' button the transcript is updated to reflect changes made.

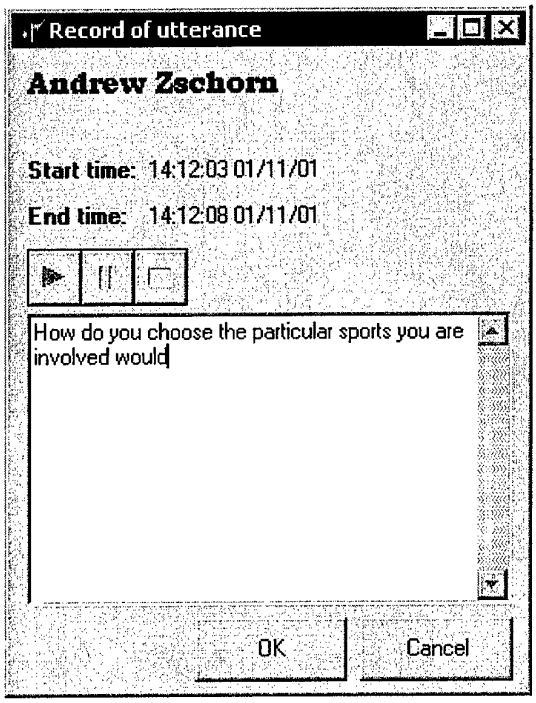


Figure 16 A screen shot of the Utterance window

4. Discussion

This section of the report discusses some of the interesting and improvable aspects of *AuTM* to date.

It should be noted that the design of the applications as presented in this report is less than ideal for real-world use. If this system were to be put into everyday use it would be better to have a single, symmetric *Client/Server* program. That is, every *AuTM* program would be capable of assuming the roles of both *Client* and *Server*, but that in any given *AuTM Session* only one of the programs would act as both, and the others would all be normal *AuTM Clients*. This would mean that the meeting moderator is able to contribute to *Sessions* just like an attendee, but that the other functions of the *Server* program would be opened-up to the attendees.

There are also some technical problems with the prototype that ought to be overcome. The most prominent of these is the problem of focus. Focus is the concept by which *MS Windows* determines which GUI component should receive keystrokes and mouse events. That is, if you are typing into a particular text box, then that is the object that is said to have focus, and no other objects have focus at the same time. Focus is a problem because of the way *NS* sends its output to the *Client* program. The speech integration mechanism that is used requires that a particular text box be chosen for *NS* to send its recognition results. This implementation is such that *NS* will only send its speech recognition results when the chosen text box has focus. So if an attendee decides to bring up a document during an *AuTM Session*, then focus will be taken from the *AuTM Client*, therefore anything that the attendee says will be ignored and not contributed to the transcript. This is a problem that needs to be solved if attendees were to use their computers for other tasks while running the *AuTM Client*. This focus issue would also need to be solved if the *Client* and *Server* programs were rolled into one. For instance, if the moderating attendee starts adding a highlight they would lose focus on their text box.

The dual-socket links are, in some respects, a bit clumsy. As each *Client* must connect to a *Server* twice, there is more room for error in trying to keep the two links 'in order' and working in harmony. For instance, if the *Server* loses track of which initial link corresponds to which second link for a *Client*, or if one of a *Client's* two links is lost, it becomes unclear whether it is truly connected or not. However, everything lost to clumsiness is overcome by the gain in ease of implementation. As messages must be sent from *Server* to *Client* and vice versa and as the message flow does not alternate from one direction to the other, having two links removes many of the problems created when using one socket connection to do all the work. Most of these problems seem to come about from the *Delphi* implementation of sockets. When relying on the standard *OnRead* and *OnWrite* events of the *TClientSocket* object and only one connection, it proved difficult to achieve reliable duplex communication. By employing two sockets, the *Client* and *Server* themselves are able to keep track of when the sockets can be written to, rather than rely solely on the events.

The *Transcript* window as presented in this report is only one of the many possible ways of displaying the transcript information. In *AuTM's* case transcripts are meant only to provide a record, not necessarily an accurate verbatim capture. This is because the attendees ought to be interacting with each other, as in a normal meeting situation, rather than with their computers. This being the case, it is still preferable to represent transcripts in a manner that is extremely easy to read and follow, especially from a greater distance than one might normally be from computer screens.

The display type shown in Figure 4 is a 'scrolling' display. In this mode, each utterance is appended to a list of previous utterances and the list is moved up. This means that the most recent utterance is always displayed on the bottom of the list. Colour is used to identify who contributed the utterance, which provides extra context for the reader (Brooks 2000). This mode has the advantages of being simple to implement and it lends itself to scrolling back through the transcript. Its major disadvantage is that the text moves every time a new utterance is added, which, it is assumed, makes it less easy to read.

Another format could be described as a 'moving-blank' display, an example of which is shown in Figure 17. Moving-blank displays are similar to the scrolling display, except that once the window fills with utterances the next is added at the top, and then it continues down until full again, rather than the list simply scrolling up. The reason it is called a moving-blank display is because when a new utterance is added, the line below it is blanked out. This blank line serves to differentiate between the newly added utterance and the existing utterances. In effect, it appears that the blank line is continually moving down the window, while the utterances themselves remain stationary until they are removed. This display has the advantage of having static, and therefore easy-to-read, text. It has the disadvantage of being difficult to implement, particularly when scrolling back through the entire list of utterances.

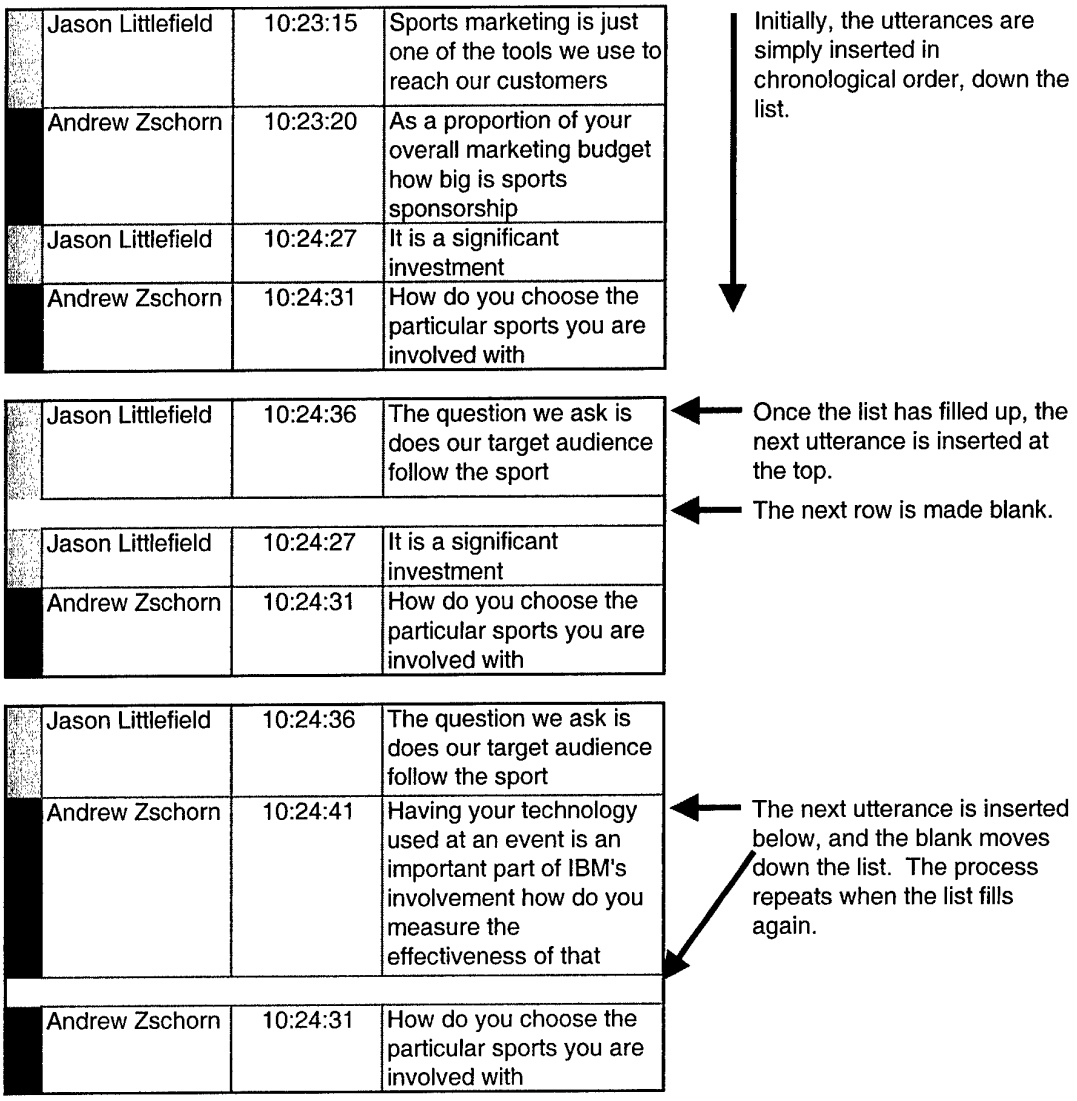


Figure 17 An example of the Moving Blank display

Brooks (2000) presents a list of characteristics of good visual presentations of speech when a text display is the only way of following a discussion. The moving blank display is preferred, with high contrast and large, simple text showing at least 50 words at any time.

A 'columnar' display would have two or three identical panels into which utterances can be added. Once one panel is full, the following utterance is added to the next panel, so one could follow the utterances as they appear much like one would read down columns of a newspaper. This mode has the advantage of having static text and an easy way of scrolling back through the utterances (flipping backwards like the pages of a book). It has the disadvantage of being difficult to implement and taking up more screen space than the other methods discussed.

There are any number of other methods for displaying this information. These three (scrolling, moving-blank and columnar) are those that appear to us to be most obvious and likely to be used. Each of these should be evaluated in order to decide which is best. Alternatively, each can be made available to the attendees, who can then choose which they prefer.

The goal of relying only on generic hardware has been met. Apart from the USB microphone adapters, only standard PCs or laptops, microphones and a LAN are required to use the *AuTM* prototype. While other transcription projects, such as one at the International Computer Science Institute (ICSI) (Janin 2001; Morgan *et al.* 2001) and a project at Carnegie Mellon University's (CMU's) Interactive Systems Laboratories (Waibel *et al.* 1998, 2001; Yu *et al.* 1998, 1999, 2000; Gross *et al.* 2000) use fewer computers, they require a special meeting room, with all the microphones and other hardware 'installed'.

Janin (2001) stated that attendees object to audio and video records being made of their meetings. He found the resistance less marked with audio records, and that attendees felt more comfortable when they were able to 'bleep out' sections of the record that they wished to remain private to the meeting. In the case of the *AuTM* prototype, the attendees can deactivate their own *NS Microphone Button* to talk 'off the record'. Using close-talking microphones means that their voice will only be picked up by another, active microphone if they speak loudly.

AuTM has not as yet been evaluated. There has been no attempt at this stage to test the proposed system out in a meeting or interview in order to assess its performance, practicality or functionality. Thus the current state of the prototype is not based upon user needs, but the assumptions of the developers. A thorough evaluation of the prototype would enable us to identify the strengths and weaknesses of *AuTM* and thus improve the system.

4.1 Speech Recognition

As *AuTM* is yet to be evaluated, the recognition performance of the system is beyond the scope of this report. The *AuTM* concept stands as-is, and the recognition accuracy is quite a separate issue. Further detail of expected speech recognition accuracy of the *AuTM* system can be found by referring to various studies conducted under the DSTO task ARM 98/288 "Application of Speech Technology and Human Computer Interaction Research to Military Messaging" and it's successor, JTW 01/092 "Human-computer interaction research, including speech and natural language approaches, in a Command Environment". These include evaluation of *NS* in noisy environments (Littlefield *et al.* 2002) and the measurement of the recognition accuracy of conversational speech versus pre-scripted speech (Broughton 2002).

However, there is room for a short discussion on the impact of less than perfect speech recognition on the usefulness of the prototype.

The decision to use a speaker-dependent ASR with headset microphones is a departure from the systems at ICSI and CMU. These systems are both pursuing the use of a

speaker-independent ASR, and avoiding or, at least seeking to do away with, the use of headset microphones.

Speaker-dependent ASRs have the advantages of being usable by speakers with a wider range of speaking styles and accents and of improving the more they are used. Also, unlike speaker-independent ASRs they typically have very large vocabularies and their grammars do not need to be predefined, meaning fluid, unstructured speech can be recognised. They have the disadvantage of needing to be trained for use by each speaker (although, in the case of *NS* version 5, the minimum required training time is only around 5 minutes). Also, a separate instance of the ASR needs to be running for each speaker.

Headset microphones have the advantages of extremely high sound quality (which aids ASR accuracy) and increased control over what is recorded. They have the disadvantage of being a nuisance and 'tying down' attendees (although this effect could be lessened by using infrared headset microphones).

The debate over the best ASR and the best microphones may be unnecessary as there may not be a need for 100% recognition accuracy at all, depending on the application of the system.

If a transcript is only used as a way of indexing, cataloguing or searching an audio record, then the recognition need not be perfect. In this case, it would be the audio that constitutes the session record, and speech-recognition would only be a convenient way to produce an index of that audio. The performance of such a system would depend on the accuracy of the speech recognition of the key utterances and words that are used to make the index.

A system that can perform a similar task to this is discussed in Hauptmann (1995) and Wactlar (2000). It uses speech recognition to label sections of recorded video. Users can then find segments of video by entering keywords, with the results being those segments whose recognition transcripts contain the keywords. Wactlar (2000) states that even in a situation where the recogniser is producing a word error rate of 50%, the recall function still performs at 85% of optimal. The high level of keyword repetition in discussion-type communication is partly responsible for the system's resistance to higher word error rates (Wactlar 2000).

Of course if, at the other extreme, the audio-recording side of *AuTM* is removed then the accuracy of the recogniser becomes extremely important. Refer to Broughton (2002) for some figures on the recognition performance of the system under these conditions.

4.2 Related Projects

There are at least three other projects underway that have goals similar to *AuTM*'s. These are being conducted at:

- BBN Technologies;
- International Computer Science Institute (ICSI) at Berkeley University; and

- Interactive Systems Laboratories (ISL) at Carnegie Mellon University.

These projects are all similar in their approach, and this approach differs significantly from *AuTM*'s. Each uses its own in-house speaker-independent ASR - all of which were initially developed for slightly different purposes, and then adapted and optimised for the meeting case - to recognise the speech of all attendees. This is the most significant difference between these projects and *AuTM*, which uses multiple instances of a COTS speaker-dependent ASR. While their approach means they have far more control over the ASRs, they are less powerful than *NS*. For instance, *NS* version 6 has a total vocabulary size in excess of 250,000 words (ScanSoft 2002), while the ASRs used in the other projects have between 34,000 and 45,000 words (Waibel *et al.* 2001; Stolcke *et al.* 2000; Colbath *et al.* 1998). The figure for *NS* is the total number of words in its active and backup dictionaries, while the others are measuring only word roots. Still, encountering Out Of Vocabulary (OOV) words is a greater problem for the other projects than for *AuTM*.

Yu *et al.* (2000) describe how to automatically search for and analyse Web pages to find words that are related to those an ASR is correctly recognising, but which are not already in its vocabulary. As words can be added to and removed from *NS* vocabularies by users, in *AuTM*'s case each attendee is responsible for maintaining their own *NS* vocabulary. To keep low the number of OOV words encountered, each attendee should edit their *NS* vocabulary so that it closely matches the vocabulary he or she uses in *AuTM Sessions*. This is most important for keywords.

Speaker dependent ASRs, such as *NS*, also have the property that speakers can correct recognition errors, and the ASR uses this information to improve its performance for that particular speaker. (It is unlikely that one would do this during a meeting, as it would be disruptive, rather one would do it off-line.) In this way, attendees can themselves improve the accuracy of the transcript. The accuracy of speaker independent ASRs cannot be improved in this way.

The BBN (Colbath *et al.* 1998) and ISL (Waibel *et al.* 1998) projects have a strong emphasis on meeting browsers. These are applications used to navigate text, audio and video records of a meeting. The BBN meeting browser is particularly advanced, with time-aligned automatic topic classification, and a search function. In contrast, *AuTM* produces simple HTML transcripts, which any web browser can view.

The ISL project can automatically summarise the transcription (Waibel *et al.* 1998), and the BBN system can do automatic topic detection (Colbath *et al.* 1998). This automated approach contrasts with that of *AuTM*, which is to leave the summarising up to a human minute taker or moderator.

Like *AuTM*, the ICSI project also uses head-worn microphones (Morgan *et al.* 2001), although they also make recordings on lapel and desk microphones with a view to substitute that less obtrusive technology in the future. ISL's system uses lapel microphones only (Yu *et al.* 2000).

Two problems brought about by using lapel or desk microphones and one ASR are speaker change detection and speaker identification. As the audio signal from

attendees' speech is picked up in more than one audio channel, it is necessary to identify which attendee is the dominant speaker by using speaker separation techniques. These issues do not arise when using head-worn microphones and separate recording channels, as in *AuTM*'s case.

Also, *AuTM*'s approach of using one close-talking microphone per attendee makes handling overlaps and interruptions a trivial matter. If two or more attendees talk at the same time, their utterances are timestamped and recorded independently. All speech information is retained as every audio record is included in the transcript, and timestamps indicate if any utterance overlapped or interrupted another.

As stated earlier, one of the features to be built into *AuTM* is that of streaming and capturing video of attendees. Of the three other projects only the ISL's can capture video of attendees. This is also used to aid speech-based speaker identification, by using colour cues and face identification.

Waibel *et al.* (2001) mention using the ISL system with attendees in remote locations. This is perhaps where *AuTM* is heading in the near future, and a task to which it is suited. Although it is not stated explicitly, the other three projects seem to be centred around a 'special' meeting room which is perhaps wired with microphones and has computers with eight-channel sound cards and so on. In contrast, *AuTM* is more mobile in nature, and its almost exclusive use of commonplace hardware and software means that it should be cheap and quick to deploy.

In summary, there are at least three other projects that have goals similar to *AuTM*'s. They have far more in common with each other than any of them have with *AuTM*. All of them use a single, in-house speaker-independent ASR compared with *AuTM*'s multiple, COTS speaker-dependent ASRs. *AuTM* is intended to be mobile and not to rely on specialist hardware, while the other three projects appear to use a purpose meeting room.

4.3 *AuTM* modified for *FOCAL*

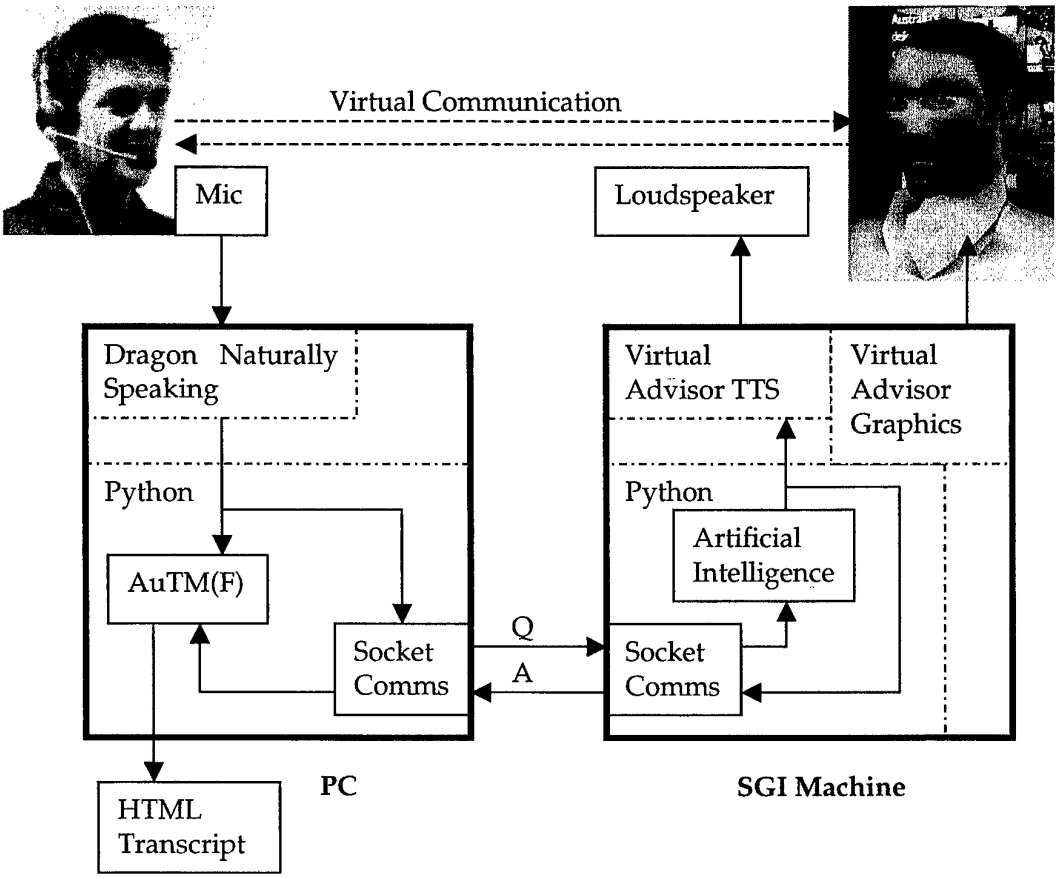
A version of *AuTM* has been developed as a concept demonstrator for the Future Operations Centre Analysis Laboratory (*FOCAL*). The current version provides the ability to log a 'Question and Answer' session between a single human operator and a Virtual Advisor.

The system developed for the *FOCAL* is a 'cut down' version of the *AuTM* system described in this report. However, it provides an output that closely represents the original system. For the *FOCAL* version, the software to perform the *AuTM* capability has been developed in the Python programming language. This came about as existing Python software provided a socket communication between *NS* running on a PC and an SGI machine. The SGI machine receives questions from the PC and provides the Virtual Advisors graphics, artificial intelligence and voice synthesis.

In this demonstrator version, the human operator's question is converted from speech to text using *NS*. This question string provides a record of the human operator's utterances for *AuTM* and is also passed via a socket connection to the Virtual Advisor

on the SGI machine. The Virtual Advisor processes the question string and an answer string is produced. This answer string is passed to a speech synthesis engine as part of the Virtual Advisor and also directly to the *AuTM* system on the PC.

The output produced by this version of *AuTM* is only available in HTML format and it provides an additional hyperlinks column for any images or movies that are presented in a response from the Virtual Advisor. As an example, the Advisor may present an image of a particular aircraft that it is presenting information on. A hyperlink to this image is stored along with the transcribed text, allowing a reader of the transcript to view any images that were presented during the conversation.



5. Future Directions

The first area of improvement to the *AuTM* system should be in overcoming its current technical problems. This includes: removing its reliance on signal-splitting cable, and therefore the USB adapters, and solving the 'focus' problems so that recognition results are obtained no matter what attendees are doing on their computers.

The most obvious direction in which to push this project is towards integrating IP conferencing and other Computer Supported Cooperative Work tools such as group decision support systems. The ability to automatically generate a permanent text record of sessions that use this kind software may make them a more effective and appealing way to work.

At present, the messages sent between *Server* and *Clients* contain ASCII and raw binary. Clearly this is insecure and needs to be changed if *AuTM* is to be used to record events that are classified. Both the messages sent during the *Session* and the storage of the final transcript need to be made secure. Encryption and password protection should probably be implemented.

While the Highlights concept provides a useful way for the moderator to distil the transcript, it requires human control. There are, conceivably, situations in which *AuTM* would be used in such a way that the Highlights summary is inadequate or non-existent. In these cases an automatic summarisation process is necessary.

Any of the information extraction processes may well be applicable in achieving this goal. The Maximal Marginal Relevance (MMR) metric, as applied in Waibel *et al.* (1998), seems to produce excellent results when measuring a human's ability to categorise a transcript given only the MMR summary. Evaluating different summarisation algorithms' performance given different types of transcripts is certainly worthy of pursuit in relation to improving the ease of browsing, scanning and storing transcripts for the long-term.

Another area of development is in applying the *AuTM* concept to transcribe pre-recorded audio. In situations in which it is infeasible for every attendee to have their own computer, it would be much preferable to use a small recording device (such as a minidisk or digital audio tape recorder) to make only an audio record of the event. *AuTM* could then be used to generate the transcript of the audio off-line.

When the development of the *AuTM* system has reached a satisfactory level, it should be evaluated. A study should gauge the usefulness, effectiveness, ease, and impact of employing *AuTM* in several areas of application, including transcribing meetings, interviews, brainstorming sessions and computer supported collaborative work. Work on the prototype so far has been carried out from a highly hypothetical viewpoint. A real end-user purpose needs to be identified in order to focus *AuTM*'s development on that particular situation.

6. Conclusions

The *AuTM* prototype described in this report demonstrates one of the possible ways of using speech recognition to automatically create records and transcripts of discussions. *AuTM* has been developed with a meeting application in mind, and as such it provides some functions that enable minutes to be produced more efficiently.

It also has the ability to record the audio of the attendee's speech, enabling human-correction of the recognised text and reviewing the transcript in a more complete and trustworthy manner. That is, an audio record provides a safety net, making up for the imperfections of the speech recognition.

The end product of the system is an HTML or *MS Word* document. This transcript includes all the attendee, agenda, highlight and utterance information of the *Session*. These formats have been chosen for their universal (or close to it) compatibility and ease of implementation.

Those areas that most deserve urgent development efforts are in a software solution to remove the reliance on extra audio hardware and integrating *AuTM* and IP conferencing software. Removing the extra audio hardware is particularly important, as one of *AuTM*'s assets is its low reliance on specialist hardware.

The prototype is based on an, as yet, unproven concept. The real-world applications are yet to be clearly defined and the prototype is yet to be rigorously tested or evaluated.

However *AuTM* does open up a wide range of possible applications and areas of further development. Several transcription tasks have been proposed, including IP conferencing, interviews, brainstorming sessions, meetings and briefings. The benefits of using a system such as *AuTM* for these purposes needs to be fully explored and evaluated.

7. Acknowledgements

We would like to thank Greg Chase of DSTO Command & Control Division for his support and funding, which allowed this year of research and development to take place.

We would also like to thank Oliver Carr of DSTO Command & Control Division for his technical help during the development of *AuTM* and his significant contribution to speech integration work in *FOCAL*.

8. References

Andrea Electronics 2002, *Andrea Electronics Online Store – USB Adapter* [Online, accessed 5 Nov. 2002]. URL: <http://www.andreaelectronics.com/Buy/ProductDesc/USB_Adapter.htm>

Brooks, C. 2000, 'Speech-To-Text Systems for Deaf, Deafened and Hard-Of-Hearing People', *Proceedings of the IEE seminar on Speech and Language Processing for Disabled and Elderly People 2000* (Ref. No. 2000/025), Institute of Electrical Engineers, London, pp. 5/1–5/4.

Broughton, M. 2002, 'Measuring the Accuracy of Commercial Automated Speech Recognition Systems During Conversational Speech', *Proceedings of the Virtual Conversational Characters Workshop, Human Factors Conference 2002* (HF2002) [Online, accessed 16 Jan. 2003], Melbourne, Australia, December 2002. URL: <http://www.vhml.org/workshops/HF2002>.

Colbath, S. and Kubala, F. 1998, 'Rough'n'Ready: A Meeting Recorder and Browser', *Proceedings of the 1998 workshop on Perceptual User Interfaces* [Online, accessed 5 Nov. 2002]. URL: <http://www.cs.ucsb.edu/conferences/PUI/PUIWorkshop98/Papers/Colbath.pdf>

Gross, R., Bett, M., Yu, H., Zhu, X., Pan, Y., Yang, J. and Waibel, A. 2000, 'Towards a Multimodal Meeting Record', *Proceedings of the 2000 IEEE International Conference on Multimedia and Expo*, Vol. III, IEEE, pp. 1593-1596.

Hauptmann, A. G. 1995, 'Speech Recognition in the Informedia™ Digital Video Library: Uses and Limitations', *Proceedings of Seventh International Conference on Tools with Artificial Intelligence*, IEEE, Herndon, USA, pp. 288-294.

Janin, A. 2001, 'Meeting Recorder' [Online, accessed 5 Nov. 2002]. URL: <http://www.icsi.berkeley.edu/ftp/global/pub/speech/papers/janin-avios2001.pdf>

Kotmel, A. 2000, 'Microsoft® SAPI Audio Objects and the *Dragon NaturallySpeaking*® SDK', Dragon Systems Inc.

Krishnamoorthy, B., Kong, J. H. and Beaumont, G. 2000, *Automated Text Extraction of Meetings*, University of Adelaide, Unpublished report.

Lernout & Hauspie 2000, *Dragon NaturallySpeaking SDK C++ Help*, Lernout & Hauspie, version 5 beta of the *Dragon NaturallySpeaking* software development kit.

Littlefield, J. and Hashemi-Sakthsari, A. 2002, *The Effects of Background Noise on the Performance of an Automatic Speech Recogniser*, DSTO Research Report DSTO-RR-0248, Command and Control Division, ISL, DSTO.

Mills, D.L. 1999, *Network Time Protocol (NTP) General Overview* [Online, accessed 5 Nov. 2002]. URL: <http://www.eecis.udel.edu/~mills/database/brief/overview/overview.pdf>

Morgan, N., Baron, D., Edwards, J., Ellis, D., Gelbart, D., Janin, A. Pfau, T., Shriberg, E. and Stolcke, A. 2001, 'The Meeting Project at ICSI', Proceedings of the Human Language Technology Conference, San Diego, USA.

Scansoft 2002, *Dragon Naturally Speaking Professional Solutions Data Sheet* [Online, accessed 15 Nov. 02] URL: <ftp://ftp.scansoft.com/pub/doc/naturallyspeaking/DNS6ProfDatasheet.pdf>

Stolcke, A., Bratt, H., Butzberger, J., Franco, H., Rao Gadde, V.R., Plauché, M., Richey, C., Shriberg, E., Sönmez, K., Weng, F. and Zheng, J. 2000, 'The SRI March 2000 Hub-5 conversational speech transcription system', *Proceedings of the NIST Speech Transcription Workshop*, College Park, USA.

Wactlar, H.G. 2000, 'Informedia – Search and Summarization in the Video Medium', *Proceedings of Imagina 2000 Conference*, Monaco.

Waibel, A., Bett, M., Finke, M. and Stiefelhagen, R. 1998, 'Meeting Browser: Tracking and Summarizing Meetings', *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, USA pp 281-286.

Waibel, A., Bett, M., Metze, F., Ries, K., Schaaf, T., Schoultz, T., Soltau, H., Yu, H. and Zechner, K. 2001, 'Advances in Automatic Meeting Record Creation and Access', *Proceeding of the International Conference on Acoustics, Speech and Signal Processing 2001*.

Yu, H., Clark, C., Malkin, R. and Waibel, A. 1998, 'Experiments in Automatic Meeting Transcription Using JRTk', *Proceedings of the 1998 IEEE International Conference of Acoustices, Speech and Signal Processing*, Vol. II, IEEE, pp 921-924.

Yu, H., Finke, M., Waibel, A. 1999, 'Progress in Automatic Meeting Transcription', *Proceedings of 6th European Conference on Speech Communication and Technology (Eurospeech-99)*, Budapest, Vol. 2, pp 695-698.

Yu, H., Tomokiyo, T., Wang, Z. and Waibel, A. 2000, 'New Developments in Automatic Meeting Transcription', *Proceedings of the International Conference Spoken Language Processing*, Beijing.

Glinert, S. 1999, 'The Final Say', *ZD Net*, 15 October [Online, accessed 7 May 2003] URL: <http://www.zdnet.com/products/stories/reviews/0,4161,2354388,00.html>

Appendix A
A Sample Transcript

Below is a sample transcript produced by the *AuTM* system during a demonstration.

Demonstration of AuTM system

The title of the record, as nominated by the moderator.

Meeting convened at 261506I October 01 (3:06PM on 26 October 2001)

The date and time the *Session* commenced, recorded automatically.

Those present:

Andrew Zschorn
Jason Littlefield

List of *Session* attendees.

Agenda:

Transcription
Interview
Interruptions
Overlap
Within
Highlights
Action Item
Motion

Agenda of the *Session*, as nominated by the moderator. These are hyperlinked to sections of the transcript (below), using timestamps to determine which utterances are related to which agenda item. In this example, the agenda items are derived from the aspects of *AuTM* we were emphasizing during a demonstration.

Highlights:

The list of *Highlights* nominated by the moderator during the *Session*.

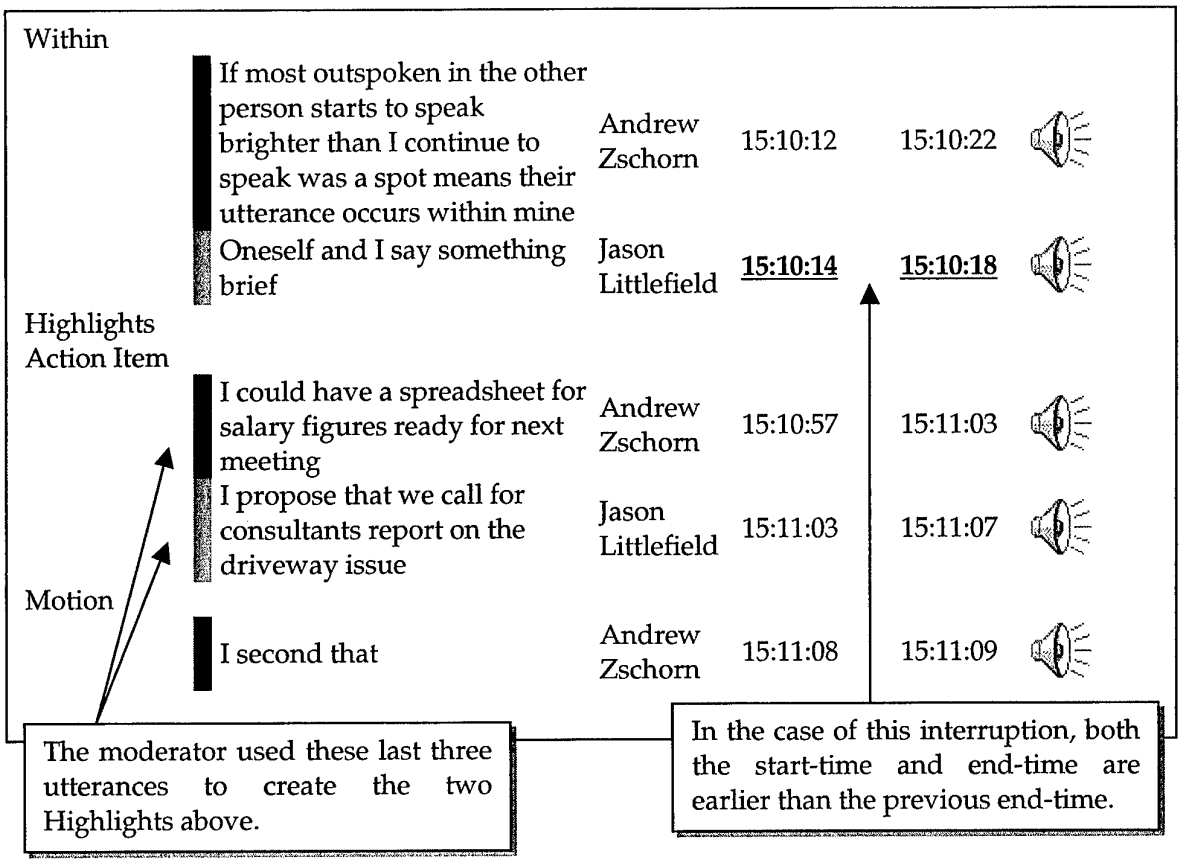
- Andrew Zschorn is responsible for ensuring the task "having a spreadsheet for salary figures ready for next meeting" is completed by 29/06/01
- The motion "I propose that we call for consultants report on the driveway issue" was proposed by Jason Littlefield and seconded by Andrew Zschorn, and carried.

Proceedings:

Agenda Item		Speaker	Started	Finished	Audio
Transcription Interview	Were valued as a company decides you with an established global profile sea in sports sponsorship	Andrew Zschorn	15:08:05	15:08:12	

	Sports marketing is just one of the tools we use. To reach customers and potential customers. Sport is a great vehicle because people are very passionate about it	Jason Littlefield	15:08:15	15:08:27	
	As a proportion of your overall marketing budget how big the sports sponsorship	Andrew Zschorn	15:08:29	15:08:34	
	The investment. And we use advertising and PR to leverage particular properties	Jason Littlefield	15:08:36	15:08:44	
	How do you choose the particular sport you are involved with	Andrew Zschorn	15:08:44	15:08:48	
	We have very strict criteria are used to evaluate opportunities	Jason Littlefield	15:08:48	15:08:54	
	Does this mean you will only go for mainstream sports such as tennis	Andrew Zschorn	15:08:55	15:08:59	
	The main criteria. Is whether it helps us reach a target audience. That they follow the sport. If the answer was no	Jason Littlefield	15:09:00	15:09:10	
	Having your technology used at an event is an important part of your involvement how the measure the effectiveness of that	Andrew Zschorn	15:09:10	15:09:17	
	We would not. There are many different ways. But we are very different from a consumer for such as a soft drink manufacturer	Jason Littlefi			
Interruptions Overlap					
	If I start to speak on the other person starts to speak and then I stopped by the other person continues	Andrew Zschorn	15:09:48	15:09:55	
	That I begin to speak before the other person has finished eating and I continue to speak up the other person has finished. Meaning out of	Jason Littlefield	**15:09:52**	15:10:05	

When an attendee interrupts another, the start-time of that utterance is highlighted.


Within
If most outspoken in the other person starts to speak brighter than I continue to speak was a spot means their utterance occurs within mine
Andrew Zschorn
15:10:12
15:10:22
Oneself and I say something brief
Jason Littlefield
15:10:14
15:10:18
Highlights
Action Item
I could have a spreadsheet for salary figures ready for next meeting
Andrew Zschorn
15:10:57
15:11:03
I propose that we call for consultants report on the driveway issue
Jason Littlefield
15:11:03
15:11:07
Motion
I second that
Andrew Zschorn
15:11:08
15:11:09
The moderator used these last three utterances to create the two Highlights above.
In the case of this interruption, both the start-time and end-time are earlier than the previous end-time.

DISTRIBUTION LIST

Transcription of Multiple Speakers Using Speaker Dependent Speech Recognition

Andrew Zschorn, Jason Littlefield, Michael Broughton,
Barry Dwyer and Ahmad Hashemi-Sakhtsari

AUSTRALIA

DEFENCE ORGANISATION

Task Sponsor

Comd DJFHQ	1
Deputy Chief Information Officer	1

S&T Program

Chief Defence Scientist	shared copy
FAS Science Policy	
AS Science Corporate Management	
Director General Science Policy Development	
Counsellor Defence Science, London	(Doc Data Sheet)
Counsellor Defence Science, Washington	(Doc Data Sheet)
Scientific Adviser Joint	1
Navy Scientific Adviser	(Doc Data Sheet & distribution list)
Scientific Adviser – Army	1
Air Force Scientific Adviser	1
Scientific Adviser to the DMO M&A	(Doc Data Sheet & distribution list)
Scientific Adviser to the DMO ELL	(Doc Data Sheet & distribution list)
Director Trials	1

Information Sciences Laboratory

Chief, Command and Control Division	(Doc Data Sheet)
Research Leader, Command and Intelligence Environments Branch	1
Research Leader Military Information Enterprise Branch	1
Research Leader Theatre Operations Analysis Branch	(Doc Data Sheet)
Head, Human Systems Integration Group	1
Task Manager, JTW 01/092	1
Greg Chase, Information Exploitation Group	1
Author(s):	
Andrew Zschorn, Human Systems Integration Group	1
Jason Littlefield, Human Systems Integration Group	1
Michael Broughton, Human Systems Integration Group	1

Barry Dwyer, Department of Computer Science,
Adelaide University 1
Ahmad Hashemi-Sakhtsari, Human Systems Integration Group1
Head Virtual Enterprises (Doc Data Sheet)
Head Systems Simulation and Assessment (Doc Data Sheet)
Head Theatre Operations Analysis (Doc Data Sheet)
Head Intelligence Analysis (Doc Data Sheet)
Head C2 Australian Theatre (Doc Data Sheet)
Head HQ Systems Experimentation (Doc Data Sheet)
Head Information Systems (Doc Data Sheet)
Head Information Exploitation (Doc Data Sheet)

DSTO Library and Archives
Library Edinburgh (1 copy and Doc Data Sheet)
Australian Archives 1

Capability Systems Division
Director General Maritime Development (Doc Data Sheet only)
Director General Land Development 1
Director General Aerospace Development (Doc Data Sheet only)
Director General Information Capability Development (Doc Data Sheet only)

Office of the Chief Information Officer
Deputy CIO (Doc Data Sheet only)
Director General Information Policy and Plans (Doc Data Sheet only)
AS Information Structures and Futures (Doc Data Sheet only)
AS Information Architecture and Management (Doc Data Sheet only)
Director General Australian Defence Simulation Office (Doc Data Sheet only)

Strategy Group
Director General Military Strategy (Doc Data Sheet only)
Director General Preparedness (Doc Data Sheet only)

HQAST
SO (ASJIC) (Doc Data Sheet only)

Army
Chief of Staff, DJFHQ, Enoggera, Qld 4051 1
DC2I, Knowledge Systems 1
ABCA National Standardisation Officer, Land Warfare
Development Sector, Puckapunyal (email Doc Data Sheet)
SO (Science), Deployable Joint Force Headquarters (DJFHQ) (L),
Enoggera QLD 1
SO (Science) - Land Headquarters (LHQ),
Victoria Barracks NSW (Doc Data Sheet & Ex Summary)

Intelligence Program

DGSTA Defence Intelligence Organisation	1
Manager, Information Centre, Defence Intelligence Organisation	1
Assistant Secretary Corporate, Defence Imagery and Geospatial Organisation	(Doc Data Sheet only)

Defence Materiel Organisation

Head Airborne Surveillance and Control	Doc Data Sheet
Head Aerospace Systems Division	Doc Data Sheet
Head Electronic Systems Division	Doc Data Sheet
Head Maritime Systems Division	Doc Data Sheet
Head Land Systems Division	Doc Data Sheet
Head Industry Division	Doc Data Sheet
Chief Joint Logistics Command	Doc Data Sheet
Management Information Systems Division	Doc Data Sheet
Head Materiel Finance	Doc Data Sheet

Defence Libraries

Library Manager, DLS-Canberra	(Doc Data Sheet only)
Library Manager, DLS - Sydney West	(Doc Data Sheet Only)

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy Library	1
Head of Aerospace and Mechanical Engineering	1
Serials Section (M list), Deakin University Library, Geelong, VIC	1
Hargrave Library, Monash University	(Doc Data Sheet only)
Librarian, Flinders University	1
David Powers, School of Informatics and Engineering, Flinders University	1

OTHER ORGANISATIONS

National Library of Australia	1
NASA (Canberra)	1

OUTSIDE AUSTRALIA

INTERNATIONAL DEFENCE INFORMATION CENTRES

US Defense Technical Information Center	2
UK Defence Research Information Centre	2
Canada Defence Scientific Information Service	1 pdf copy
NZ Defence Information Centre	1

ABSTRACTING AND INFORMATION ORGANISATIONS

Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
Materials Information, Cambridge Scientific Abstracts, US	1
Documents Librarian, The Center for Research Libraries, US	1

SPARES (5 copies)

Total number of copies: **44**

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

Transcription of Multiple Speakers Using Speaker Dependent Speech Recognition

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

Andrew Zschorn, Jason S. Littlefield, Michael Broughton, Barry Dwyer and Ahmad Hashemi-Sakhtsari

5. CORPORATE AUTHOR

Information Sciences Laboratory
PO Box 1500
Edinburgh South Australia 5111 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TR-1498	AR-012-901	Technical Report	September 2003

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
E9505-25-50	JTW 01/092	COMD DJFHQ & DCIO	36	22

13. URL on the World Wide Web

http://www.dsto.defence.gov.au/corporate/reports/DSTO-TR-1498.pdf

14. RELEASE AUTHORITY

Chief, Command and Control Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DEFTEST DESCRIPTORS

Speech recognition
Computational linguistics

19. ABSTRACT

Speech recognition can potentially aid in producing transcripts of discussions, interviews, meetings, and other collaborative efforts. This report presents a concept demonstrator of an automatic transcription system that produces text and audio records using speaker dependent speech recognition. It is defined and discussed in terms of how it works, how users operate it, and its similarities and differences with other transcription systems.