# Developing a Videomail Service for iLanga, an Asterisk Based IP PBX

Zelalem S. Shibeshi[1], Alfredo Terzoli[2], Karen Bradshaw[2]
Department of Computer Science, Rhodes University
Grahamstown, South Africa
+27 46 603 8291
[1] zelalems@rucus.ru.ac.za
[2] {A.Terzoli, K.Bradshaw}@ru.ac.za

**Abstract— The requirement for receiving a message when a person is unable to answer his/her phone has existed since the introduction of telecom services. This has led to the development of voicemail systems. With the advent of VoIP, video communication, which was once thought to be a great challenge, has also become a reality. However, not all VoIP systems provide videomail service. The current stable version of Asterisk, an open-source IP PBX, does not natively include video recording and playback functionality (the main functions of a videomail service). As a result, an application was developed that can record and playback video messages in mp4 format. We extended this application to provide a full fledged videomail service for Asterisk. This paper discusses this extension and its integration in iLanga, the Asterisk-based IP PBX which is in use by Computer Science postgraduate students.**

**Index Terms — Asterisk, mp4, videomail, video-oriented service.**

## I.INTRODUCTION

One of the services that users would be keen for Telecom operators to provide is a messaging service that is able to receive messages when calls cannot be answered and then to play it back on demand. Messaging in the context of VoIP means the recording and delivery of either voice or video messages. Asterisk, an open source IP PBX (Private Branch Exchange) that gives all the functionality of high-end business telephone service [1], including audio and video calls, provides a native voicemail service. Asterisk's voicemail service also includes the sending of an e-mail to a user's inbox by attaching the recorded voice message. However, it does not support videomail natively.

The Computer Science Department at Rhodes University has been utilising Asterisk together with other VoIP and non-telephony software for communication activities for a long time. This integrated communication system, called iLanga, was developed by the VoIP research group at the Department by implementing various VoIP and non-telephony technologies to extend the capabilities of Asterisk [2]. iLanga, in addition to Asterisk, uses SER (the SIP Express Router) as a SIP proxy and registrar, OpenGK, an implementation of the H.323 Gatekeeper, and a MySQL database to provide a comprehensive communication service. One of the main advantages of iLanga is that it makes use of user profiles, allowing user details to be registered, and devices to be allocated. On the other hand, with the aim of providing an easy access to the above mentioned communication system, a flash-based management and control web interface, has also been developed and integrated into iLanga [3]. The web interface allows users to easily set up their profile and communication devices, access their voicemails, and update their credit information for the prepaid modules.

Currently the Convergence research group (formerly called the VoIP research group) is engaged, amongst other activities, with the development of various video-oriented services. As a first attempt, we have developed videomail systems in two separate environments. In this paper, the videomail service for the iLanga/Asterisk environment is discussed. The system gives complete video recording and playback functionality. The other videomail system that is under development is based on a pure IMS platform and at present, offers only a video message delivery service. This system assumes that video messages are already stored in users' mailboxes and does the delivery when the user contacts the system. Currently, development work is being done to include the video recording capability to this system so as to make it a complete and usable system. The focus of this paper is to present the development activity of the videomail service developed for Asterisk.

## II.RELATED WORK

The number of video oriented systems developed especially on the Internet is increasing rapidly. The systems developed so far give different type of video oriented services, including video chatting, video conferencing, and videomail. As the need for videomail system in Asterisk was evident, with the idea of finding a videomail system to plug into Asterisk, we tried to identify the available video oriented systems. We found five systems that give various types of video oriented services [4-7], but based on our preliminary assessment, none of them could fulfil the requirement that we were looking for, a videomail system that can be pluggable and understands SIP protocol.

In fact, only the systems specified in [4], [7] and [8] have videomail service capability. But, all of them are web-based and they don't talk SIP, the protocol that Asterisk could use to inform the availability of video message to a plug-in videomail system. The type of codec that the systems support is also not specified in all cases. As a result, because

This work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom, Comverse, StorTech, Tellabs, Amatole Telecom Services, Mars Technologies, Bright Ideas 39, and THRIP.

of these and other factors, we disregard the possibility of using a third party videomail system to integrate with Asterisk and continued with the development of our own open source videomail system, which includes various mechanism of accessing the video message.

Ref. [9], on the other hand, showed how Asterisk can be integrated in IMS and be used as a Media Server for Video Conference service provisioning. The videomail service discussed in this paper could be used with such system since recording media is one of the requirements of conferencing applications.

The next section gives a brief overview of the video capability of Asterisk.

## III. Asterisk's Video Capability

As described in [10], video support for Asterisk even in the latest stable release[2], is in its infancy and has various problems related to video codec and advanced video attributes (such as frame rate and image size) negotiation, video playback and video transcoding. Despite these limitations, users can make video calls by changing certain configuration settings on both the server and the client side.

The basic configuration setting required on the server side is an indication that Asterisk should provide video support. This is done by inserting the statement "videosupport = yes" in the configuration file, "sip.conf". In addition to this, for each user/peer, supported codec entries for video need to be specified in the context defined for the user in another configuration file, "extensions.conf".

Asterisk supports most of the common codecs file formats for audio communication. However, with regard to video communication, it only supports H.261, H.263 (and 263+) and H.264 codecs, but does not support any video file format. H.263+ and H.264 codecs have been supported in Asterisk since release 1.4. However, H.261 and H.263 were present natively in the technology even before the 1.4 release.

Video communication also requires certain changes to the configuration settings on the client side. Depending on the type of client used, the client needs to be configured to enable video communication and also the supported video codecs need to be selected. Unless video is explicitly enabled, most clients (softphones) allow only audio communication by default. In addition, some clients do not start sending video automatically. Thus, users may need to instruct the client to start sending video manually (this can be done simply by pressing an appropriate button). X-Lite, which is used as the client software in this work, has to be instructed to do this when the phone is connected. On the other hand, if the client is used in a Linux environment, the video capture device should also support the Linux Video Plug-in API (V4L or V4L2). A list of webcams and devices compatible with the above video plug-in is given in [11].

Once the above configurations have been set on both the server and the client side, users can communicate using video. However, the videomail functionality is still missing because Asterisk does not correctly support video recording and playback functions, which are the basic functions required from a VoIP system to implement videomail. Although Asterisk can dump contents of the RTP packets including some timing information in .h263, .h263p, and .h264 files, this cannot be played back or streamed properly as ordinary video content [10]. In other words, Asterisk does not use a proper file format to store video contents. As a result, an initiative has been undertaken to develop a pluggable application that can record and playback video in mp4 video file format. One important feature of the mp4 file format is that it can embed any kind of multimedia data. This video recording and playback application, called *app_mp4*, is an important achievement in making Asterisk a competitive infrastructure for video communication in a business environment. This application is currently used by the Asterisk Video Community to develop various video related services. The videomail system described in this paper has also been developed by utilising this application. The following section describes the app_mp4 application.

## IV. The app_mp4 application

The *app_mp4* video recording and playback application has been developed to provide video recording and playback functionality in Asterisk. The application is still in its infancy and as such, there is not sufficient documentation on the setup and use of the application [12]. Consequently, setting up the application is itself, a big challenge. The application depends on the development libraries of the MPEG4IP package [13] and thus, one needs to install the MPEG4IP development libraries and also the other multimedia packages on which MPEG4IP depends before installing and configuring the *app_mp4* application.

There is also no clear information on how to set up clients for use in the application. The *app_mp4* application only works with ulaw and alaw audio codecs, and with regards video codecs, the application supports H.263 and H.263+. Other video and audio codecs must not be selected (enabled) in the client to allow the application to work properly. Depending on the type of selected video codec, it is also important to set up the minimum bandwidth.

Once the application has been set up and integrated with Asterisk, one can use its two functions, namely the *mp4play* and *mp4save* for playing back and recording mp4 files, respectively. This is done by calling the two functions at the appropriate place in the file "extensions.conf".

Both the *mp4play* and *mp4save* modules require the filename of the video message to be supplied as a parameter. The modules will then use the file name to save or playback the video message. This implies that if one wants to create new messages, he/she has to change the file name and location manually each time by editing the "extensions.conf" file before calling these functions. Otherwise, the application overwrites the video messages as new messages come in. It is also not possible to tell the application to save video messages to a user's specific message box for the same reason mentioned above.

This resulted in the need for an archiving system (application) that automatically identifies a user's message box and also the appropriate file name to store the video message in the appropriate mail box. For this purpose an Archiving system was developed and integrated with Asterisk to make the *app_mp4* application a full-fledged videomail system. The following section explains how the videomail system was developed.

[2] The current stable release is 1.4.23.

## V. THE VIDEOMAIL SYSTEM

A messaging service can be developed either in a distributed (in the client) or centralized mode. In this regard, some clients offer videomail service in the client. However, the problem with such type of distributed system is that when a person has to go somewhere and cannot take his/her phone with him/her, he/she will not be able to access his/her messages. As a result, the preferred approach is to use the classic centralized messaging service, where a user can access his/her messages from anywhere using the communication facility that a centralized messaging system provides. However, there is also one problem even with centralized system. The user has to use the prescribed access technology to retrieve the video message. In this regard, we have tried to make our system access agnostic and give the user the possibility of using all kind of accessing techniques.

The videomail system under discussion has been developed as a centralized videomail system and provides the recording and playback of video messages to iLanga users. As is common in messaging services, the videomail system also offers users the facility to record their own greeting messages which can be used in the event that they are unable to answer their phones. In order to give users a seamless communication interface, the videomail system has been integrated with the existing communication system, iLanga.

The benefit of this videomail system is that users are not tied to any specific type of client to access their messages and can use SIP-enabled devices, network-enabled media players, or the iLanga web interface.

The videomail system has been developed using various tools and technologies that are integrated with Asterisk. Figure 1 depicts the main technologies utilised in this system.

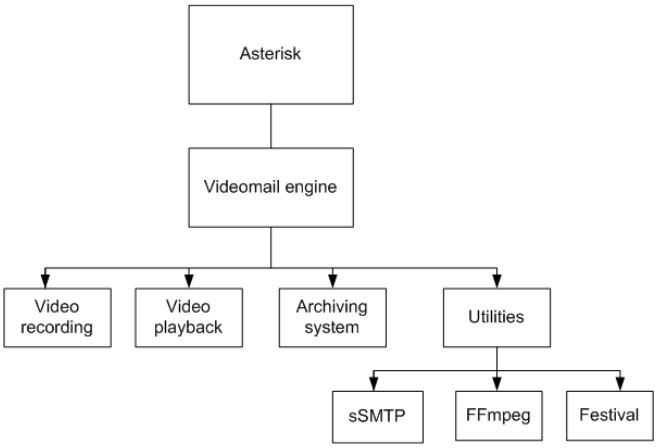


*Figure 1. Components of the videomail system*

As can be clearly seen from this figure, the videomail system is composed of six different modules which collectively called the *videomail engine* and works together with Asterisk to provide the required videomail service. Of the six components, the major component is the *app_mp4* application, which as mentioned previously gives the video recording and playback service in the mp4 file format. However, to give *app_mp4* a videomail capability, an archiving system, *app_archiver* (henceforth referred to as 'the Archiver'), was designed and developed using Asterisk's application interface API.

For the purpose of alerting users about new videomails, the Archiver also uses the sSMTP (Simple SMTP) mail engine to send mail to users. This will allow users to access their video messages using media players.

Festival, a speech synthesis engine, and FFmpeg that provides video streaming and converting functionality, are also used to assist other components of the system.

As a first step to the preparation of the videomail system, a script has been incorporated in iLanga to create a videomail box for each user. Accordingly, a videomail box is created for each user in the '/var/mail/videos/' folder. The videomail box has two sub folders, one named "Inbox" (for storing video messages) and the other named "Archive" (for storing viewed messages). For example, a user who has an extension number 1000 has the mailbox, /var/mail/videos/1000/Inbox.

The following sections discuss how the videomail system is integrated.

### *A. The Archiver*

The Archiver was developed to assist the *app_mp4* application and is implemented using the C programming language. It implements some of the Linux APIs. As mentioned above, the major task of the Archiver is to locate a user's mail box and also the appropriate file name that the message should be saved into so as to avoid overwriting new video messages. The archiving system also moves viewed video messages into another folder designed for archiving purposes after users have viewed the messages.

To accomplish the above mentioned functions, five main modules, namely *getNextFileName*, *getGreetingMessage*, *getMessage*, *moveMessage*, and *sendMail* are available in the Archiver.

Once an application is compiled and integrated into Asterisk, it is easy to instantiate it from within Asterisk. This is done by calling its functions at the appropriate place in the Asterisk configuration file, "extensions.conf". Accordingly, Asterisk is configured to initiate the videomail service if users do not answer their calls within 20 seconds.

### *B. Playing the Greeting Message*

If a user does not respond to a call in 20 seconds, Asterisk first plays a greeting message by calling the Archiver and *app_mp4* application one after the other. The greeting message could be a general or user specific greeting message, depending on whether the user has pre-recorded his own greeting message. The recording of the greeting message is described later.

Asterisk first calls the Archiver to identify the user specific greeting message. The Archiver, using the *getGreetingMessage* function, identifies this and sends the file name (if there is one) containing the greeting message to Asterisk. If there is a user specific greeting message Asterisk will pass the filename to the *mp4play* function of the *app_mp4* application to play the greeting message to the caller. Otherwise, Asterisk passes the general greeting message to the *app_mp4* application to play the general greeting message. Once this is done, the recording of the video message continues as follows.

## VI. RECORDING THE VIDEO MESSAGE

As users may have different video messages, each video

message needs to be named uniquely to avoid the problem of video message overwriting that currently happens in the *app_mp4* application. For this purpose, a unique file naming feature is implemented in the Archiving system. A file named *nextFileNo,* which is used to identify the next file number, is created and stored inside each mailbox.

Asterisk is designed in such a way that when it invokes an application from within the "extensions.conf" file, it supplies the current call information to the application that it invokes through a channel variable. The Archiver has also been designed with this idea in mind. The channel variable holds information such as the callee's extension (which, in our case, is be used to identify the mailbox) and the caller's id (again used to label the message as described above).

Accordingly, when invoked by Asterisk through *getNextFileName* function, the Archiver gets the callee's extensions from the channel variable to identify the user's message box. It then opens the folder of the user, and reads in the last "file number" from "*nextFileNo*" file. It records the file number and makes the necessary modification to the "*nextFileNo*" file. Because of the security system built into Linux, a program can only write to those files owned by the program. Therefore, to allow the Archiver to write to these files, its SUID bit was set as root before being integrated into Asterisk. This gives the Archiver root privileges thereby allowing it to write to any folder.

Once the Archiver finishes recording the file number, it forms the proper filename and path and uses Asterisk's API to pass the result to Asterisk's variable defined and included into "extensions.conf" file for this purpose. When control passes to Asterisk, Asterisk continues to initiate the *mp4save* module, which then uses the variable that the Archiver modified to start recording the video message in the appropriate location.

To make the system user friendly, a particular extension has also been configured for users to record their own greeting message. The greeting message, as mentioned previously, is used by the system when users are not able to answer their call. This message is stored in their mailbox. Similar to video message recording, the *app_mp4* application together with the Archiver is used to record the greeting message.

After the video message is recorded, the Archiver collects the e-mail address of the user and also formulates the body of the message by supplying the URL of the video message that users can use to stream the video message and supply the data to sSMTP mail transfer engine, which then do the actual sending of the e-mail message.

The following section presents how the videomail system is configured to provide video messages to users.

## VII. ACCESSING THE VIDEO MESSAGE

The videomail system is developed in such a way that it allows users to access their messages using a multitude of technologies including a SIP-enabled phone, media player, the iLanga web interface or even a web browser. The following sections show how the Archiving system is configured to allow users access to their video messages using the various access technologies.

### *A.Using a SIP-enabled client*

With access to a SIP-enabled phone, a user can call a particular extension configured for this purpose to access his/her videomail. Asterisk is configured to initiate the *getMessage* function of the Archiver when the above mentioned extension is called. The *getMessage* function is used to identify the user's mail folder and also the number of video files the user has in his/her mailbox. Accordingly, *getMessage,* spawns the user's Inbox to see the number of video files and also reads the last "file number" so that it can start playing the earlier messages. It then returns these values to Asterisk. If the user does not have any messages, zero is returned. Accordingly, Asterisk is configured to formulate a text message about the status of the user's mail box and then vocalise the text message to the user using the text to speech synthesizer, Festival [14]. Asterisk then plays the video messages one by one until all messages have been presented to the user using the *mp4_play* module. Once a video message has been played, the system calls the *moveMessage* module of the Archiver to move it to the Archiving folder.

### *B.Using Media Player*

As mentioned previously, Asterisk has a facility to send an audio message as an attachment to users' email address. However, videomails are larger in size and it is neither easy nor advisable to attach video messages to an e-mail. But, with the help of a streaming server, users can have their messages streamed to them and all what they need is a URL that contains the location of the video message and the streaming server. They can then get their messages using any media player capable of accessing network files (streaming server) or even through a browser.

The videomail system has been configured to send an e-mail that contains the URL of the video message to users. The videomail engine queries the MySQL database to obtain the user's e-mail address, after which, it formulates a message and uses the sSMTP mail agent, to send the mail to the user's e-mail address.

Thus, where users do not have access to a SIP-enabled phone to call the PBX, they can use a media player to get their message from a Media Server configured for this purpose. The Darwin Streaming Server, an open source Media Server [15] has been configured to provide this streaming service. Users can use the URL sent via e-mail by the videomail system to contact the streaming server and stream their messages using any media player that supports the RTSP streaming protocol over the network. They can even use an HTTP protocol and stream their messages using a browser that has a plugged-in media player.

### *C.Using the iLanga web interface*

Some staffs and postgraduate students at Computer Science Department have been using the flash-based web interface of iLanga to access their voicemail message. The page that provides this access was modified so as to let users access their videomail. The following figure shows the modified user interface of the VOICEMAIL page of iLanga.

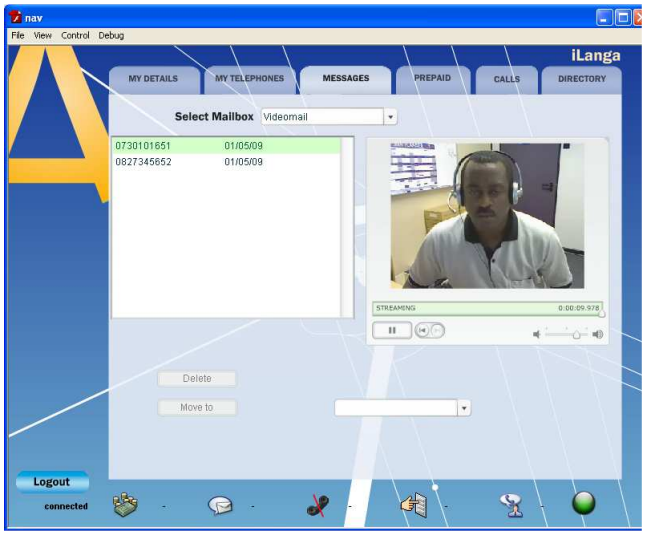


*Figure 2: The modified iLanga web interface*

The VOICEMAIL tab of the web interface was changed to "MESSAGES" to reflect its content, which now includes both voice and video messages. Accordingly, users have to choose "videomail" under the "Type of message" combo box to receive their video messages. Once a user choose the videomail option, iLanga, using a PERL program, collects and presents list of video messages to users to choose and play. The lists of video messages include the caller details and the date the message was recorded.

Users then choose the video message they want to play and start viewing it using the integrated flash video player. As Adobe Flash does not support the video codecs that the *app_mp4* application uses to create the video messages, the video files has to be converted into a format that is supported by Flash. The most recent release of Flash Player supports the playback of video files that is encoded with H.264 video codec in addition to the proprietary flash video format. Therefore, in order for users to use the iLanga web interface to access and play their video messages the video messages have to be converted into either an mp4 file with H.264 codec or into flash video file. The videomail system uses, a video conversion tool, called FFmpeg[11], to convert the video messages into flash video formats and store them in a different folder. Asterisk is configured to initiate FFmpeg to record a video message and store it in the "Flash Files" folder immediately after the recording has been done. iLanga is configured to access this folder, which resides within each user's Inbox folder, when instantiated.

## VIII. Future Work.

The videomail system, in addition to giving the intended service for users can also be extended and used to develop other video oriented services. One particular area where this can be helpful is eLearning. Ref. [17] shows how the *app_mp4* application together with other applications such as the Archiver can be used to develop an eLearning system. This is one area that needs further investigation.

Currently, the app_mp4 application can record and playback video messages using the H.263 and H.263+ video codecs. However, as presented in [18], as of 2008, the H.264 video codec is a more recent block-oriented motion-compensation-based codec standard developed by the main standard bodies, ITU-T Video Coding Experts Group (VCEG), the ISO/IEC Moving Picture Experts Group (MPEG), and the Joint Video Team (JVT). As a result, support for the H.264 codec in the videomail system is of paramount importance. Therefore, an effort will be made to include H.264 support in the videomail system.

In case users are away from their work area and are at a place where they do not have access to a SIP-enabled phone or network-enabled media player, they will not be able to view their video messages. As a result, it would beneficial to provide them with a facility that allows them to access their message using a different technology. This can be achieved by developing a web based media player that they can use via a web browser. In this regard, an initiative has been made by the Convergence Group to develop a JMF based media player applet. Currently, the applet is used to capture and play back video locally. This can be extended so that users can use the applet to access the media server from a remote location.

## IX. Conclusion

In this paper, we have presented an easy to use open-source videomail system developed for Asterisk environment. A videomail system, as any messaging system, requires the recording and playback of messages using users' mailboxes. In this regard, the video recording and playback application developed to work with Asterisk was extended to serve this purpose. The resulting videomail system provides all the functionality expected from a videomail system. One important feature included in this system is support for users to use various technologies to access their video messages. The system can also be modified or integrated with other systems to give a broader service, such as eLearning.

## References

[1] Jim Van Meggelen, Jared Smith and Leif Madsen, 2005, "ASTERISK : The Future of Telephony," O'Reilly, 2005.

[2] Penton, J. B. and A. Terzoli. "iLanga: A Next Generation VOIP Based TDM-Enabled PBX". In SATNAC Conference Proceedings (2004).

[3] Jonathan Hitchcock, "Decorating ASTERISK: Experiments in Service Creation for a Multi-protocol Telephony Environment Using Open Source Tools", Master's Thesis, Rhodes University, March 2006.

[4] TokBox - Free Video Chat and Video Mail, Available Online, URL: http://www.tokbox.com/

[5] Video Streaming and Conference Software , Open Source Software, Live broadcasting Webcasting, Available Online, URL: http://www.vmukti.com/

[6] *Dimdim* web conferencing that just works. Available Online, URL: http://www.dimdim.com/

[7] Winkball – Communicate Happiness. Available Online, URL: http://www.winkball.com/

[8] *Eyejot* is an easy way to send video. Available Online, URL: http://www.eyejot.com/

[9] A. Buono, T. Castaldi, L. Miniero, and S. P. Romano, "Design and Implementation of an Open Source IMS Enabled Conferencing Architecture", NEW2AN 2007, LNCS 4712, pp. 468–479, 2007._Springer-Verlag Berlin Heidelberg 2007

[10] voip-info.org. “Asterisk Video”, Available Online, URL: http://www.voip-info.org/wiki/view/Asterisk+video
[11] Linux UVC driver and tools, Available Online, URL: http://linux-uvc.berlios.de/
[12] Sergio Garcia Murillo, “Asterisk Video Resources”, Available Online, URL:http://sip.fontventa.com/content/view/12/41/
[13] MPEG4IP: Open Source, Open Standards, Open Streaming, Available Online, URL: http://mpeg4ip.sourceforge.net/
[14] The Festival Speech Synthesis System, Available Online, URL: http://www.cstr.ed.ac.uk/projects/festival/
[15] Darwin, “Open Source Streaming Server,” Available Online, URL: http://developer.apple.com/opensource/server/streaming.
[16] FFMPEG, Available Online, URL:http://www.ffmpeg.org/
[17] Mosiuoa Tsietsi, Zelalem Shibeshi, Alfredo Terzoli and George Wells. “A Telephony Driven Framework For eLearning Using Open Protocols and Open Source Software” (Accepted for publication), The ITU-T Kaleidoscope Events, 2009.
[18]H.264/MPEG-4 AVC, Available Online, URL: http://en.wikipedia.org/wiki/H264

**Mr. Zelalem S. Shibeshi** holds an MSc in Information Science, Diploma in Computer Science and BSc in Physics, all from Addis Ababa University, Ethiopia, and is currently working toward his PhD in the Computer Science Department at Rhodes University.

**Prof. Alfredo Terzoli** is a Professor and Head of the Centre of Excellence in the Computer Science Department at Rhodes University.

**Dr. Karen Bradshaw** is a Senior Lecturer in the Computer Science Department at Rhodes University.