Operating systems for the multimedia office of the future

J. Dospisil

Monash University, McMahon's Rd. Frankston, Vic. 3199, Australia

Phone: 61 3 9904 4135, Fax: 61 3 9904 4124

e-mail: jdospisi@fcit-fl.fcit.monash.edu.au

E. Kendall

Computer Systems Engineering, Royal Melbourne Institute of Technology, City Campus

GPO Box 2476V, Melbourne, Vic. 3001, Australia

Phone: 61 3 9660 5305, Fax: 61 3 9660 5340

e-mail: kendall@rmit.edu.au

T. Polgar

ISSC Australia, 60 City Rd. Southgate, Vic, 3006, Australia

e-mail: tpolgar@sydvm1.vnet.ibm.com

Abstract

Recent developments in distributed multimedia technology and in the integration of variety of media have the potential to change the nature of dissemination of information. However, it is important to ensure that developers are provided with proper frameworks and system services which help them to develop new applications efficiently. The capability to deliver continuous media to the workstation and meet its real-time processing requirements is now recognised as a central element of future distributed office applications. The objective of this paper is to survey the trends in support for continuous media processing in distributed environments. In particular, we look at the emerging abstractions in operating systems to ensure predictable response times for delivering continuous media at the desktop. In conclusion, we attempt to derive a set of possible directions for future multimedia operating systems.

Keywords

Operating System, Quality of Service, continuous media, reserve

1 INTRODUCTION

It is already recognised that office applications of the future will be heavily reliant on distributed multimedia systems. The user requires uncomplicated access to a great amount of information in real time, in a variety of formats and languages, and in different geographic locations. In addition, the office worker needs to be able to cooperate with other co-workers, sharing documents. From the computational and engineering point of view there are two important aspects of the future office: the availability of multiple media types, and the ability to manipulate them efficiently within a wide information space and with a predetermined **Quality of Service (QoS)**. In our research, we have identified three interleaving components which have to be considered in order to meet the two requirements of uncomplicated access and maintenance of acceptable *QoS*. These components include 1) an operating system with the capability to process media data with timing constraints (eg. Bollella, (1995) Coulson (1991), Williams (1992), Arbab (1993)) high level powerful constructs for defining a whole range of synchronisation constraints (eg. Staehli (1994), Gibbs (1991)), and Berra (1992)) reliable network transport services for moving time-constrained media data (Hoepner (1991), Williams (1992), Anderson (1990)).

In this paper, we concentrate on advances in operating system (*OS*) design which may lead to improvements in processing time-constrained media data. The paper is organised as follows: Firstly, we discuss the aspects of multimedia applications. In section 2.2 we outline a set of temporal processing requirements. In section three, we overview the emerging abstractions in *OS* to support predictable processing times for continuous media. In section four, we present the summary of technologies for the *OS* of the office of future.

2 MULTIMEDIA AND MULTIMEDIA APPLICATIONS

Throughout recent times there has been development of a variety of media types as well as their integration into complex documents. The office paradigm has moved from a simple voice and paper information representation to complex multimedia presentations. The integration requirements resulted in further requirements for precise synchronisation of multiple media presented as a complex document on the user's workstation. There is a wide range of emerging applications which combine information sources such as high quality digital audio and video, graphics, and images into elaborate precisely scheduled presentations. There are also new classes of applications currently emerging such as collaborative environments and collaborative editors.

The requirement to support multimedia in distributed and collaborative applications has raised a number of issues and discussions. One of the most important issues is that of synchronisation. Multimedia applications need sophisticated support for presentation scheduling (Hoepner (1991)) as well as for maintaining real-time processing of continuous media (Coulson (1992)).

2.1 Continuous media

Video and audio data are compressed and then before the presentation they are processed into their visual presentation. Two common processing paradigms are used: a stream model, and a time-advance model. The processing must take place in any paradigm.

Stream and time advance paradigms

A stream is a time-ordered flow of data from a source to one or more sinks. This paradigm is typically used with large data sets and continuous real-time data. Streams are sometimes likened to a hardware connection of parts. The only difference between hardware and software processing is the speed. The software components are implemented as a set of buffers which cannot process data with anything close to the speed of light. Therefore, the total delay from the source to sink can be substantial. The primary advantage of this paradigm is its ability to *connect* input (commonly known as source) and output (known as sink) streams dynamically. The combination of software and hardware components at the conceptual level is also possible. Typically, it is implemented as a set of buffers (see for example Sync/Stream Subsystem implemented in IBM's MMPM/2 for OS/2).

With the time-advance paradigm, data are computed in presentation order but ahead of real time. This ahead of time processing allows for data to be buffered, timestamped, and then output with low latency. The processor contention or preemption is leveled by the size of buffers; therefore if the source falls behind real-time the sink can still output precomputed data from its buffers, thus minimising jitter. A comparison of the stream and time advance paradigms can be found in Dannenberg (1994).

Distributed multimedia applications

A good survey of distributed multimedia applications and their processing requirements is presented in Berra (1992) and Williams (1992). Distributed multimedia applications typically incorporate digital audio and video which is delivered from a server to a client workstation over a network connection. Processing of continuous media either locally or over a network requires predictable response times throughout the entire end-to-end path. The major components which impose processing delays are the network, communication protocols, Audio/Visual System, and the *operating system (OS)* (Figure 1)

Figure 1 The end-to-end path of distributed multimedia applications.

Current commercially available OS (for example UNIX, MS-Windows, Windows NT, OS/2) are unable to support smooth processing of time critical data such as digital audio and video. The reason is the rigid real time processing requirements of so called continuous media. For example, throughput and low latency requirements are to

process a frame every 30 or 40 ms (NTSC or PAL) with no delay.

Some conceptual views of multimedia authoring

The ability to combine, modify, and synthesise media data in real-time is a difficult task to support within a conceptual framework. The precise orchestration of multimedia presentations depends on real-time utilities as well as on the authoring tools and their expressive power. The common view is to provide the programming environment first and then build a visual interface over a toolkit (or a set

of Application Programmer Interfaces or APIs). The modelling approaches range from providing a full visual environment for design and generation of applications to a simple object oriented (OO) model, toolkit and language eg. C++. Although a comprehensive survey of multimedia models is out of the scope of this paper, some examples of object oriented models are in Arbab (1993), Gibbs (1991), and Rossum (1993) to illustrate the current developments. A good survey of complex authoring support tools is in Hardman (1995).

The trend is to create easy to use frameworks which support visual authoring. In our view, the main deficiency of these models is the lack of sufficient expressive power to incorporate temporal constraints of continuous media data and the means to dictate the desired *QoS* expected from the underlying support layers. Furthermore, the coordination between the multiple streams of media data¹ and asynchronous events does not seem to be easy to incorporate into a simple programming model as suggested for example in Gibbs (1991).

Computer-participative multimedia applications

Computer-participative multimedia applications not only process live multimedia sources; they also perform analysis on their audio and video data input and take actions based on the analysis (Christopher (1994)). An example of such a system is a program that is capable of analysing television news and maintain online database of stories organised by subject. The computer here is an active participant in the processing of audio and video data.

2.2 Temporal processing requirements for continuous media

Multimedia such as live digital video and audio processing should coexist with conventional applications. The user expects to watch CNN news in one window while preparing lecture notes in another window. This requires a system which can support conventional and real-time applications without restricting either one. The ultimate goal is to provide a computational model which would support the emerging set of requirements recognised in Jeffay (1995), Mercer (1993), Anderson (1990):

1. *The application must execute deterministically.* This implies that the system must provide predictable processing times for those active objects which are time constrained.
2. *Adaptive allocation of resources.* Since the amount of resources which could be allocated to a process is limited and the resources must be shared among applications, the application should have a means of dynamically adjusting its requested services. Therefore, the application must have knowledge of the actual system performance and be able to adjust requested resources according to *QoS* parameters.
3. *High level model of QoS.* *QoS* is a set of parameters which reflects the user's desired quality of presentation. The specification of these parameters as the system concept at the application level is seen as a better conceptual solution since it enables the creation of coherent conceptual development frameworks.

Each part of the end-to-end path (see Fig. 1) may become the processing bottleneck. Our hypothesis is that the improvement in multimedia real-time processing must be in tight cooperation with the application layer (in particular providing high level model and semantics for *QoS*

¹ Our experience is based on the OS/2 MPM/2 example which seems to represent generic stream model

abstraction) and with the layers below. Yet the cooperation must be built into an abstraction which would not restrict the application's portability over multiple platforms. Furthermore, there is a need to cooperatively handle time-critical multimedia applications with conventional applications.

3 REAL-TIME OPERATING SYSTEMS TO MEET MULTIMEDIA REQUIREMENTS

Conventional operating systems provide a convenient environment for general data processing but they cannot guarantee real-time deadlines for continuous media data. With the perspective of the future multimedia office, the operating system must be designed to accommodate both processing of temporally constrained data as well as conventional tasks while maintaining acceptable *QoS* primarily in terms of predictable response times. We have surveyed five different approaches:

1. Predictable OS protocol processing [Mach 3.0] - section 3.1
2. Coexistence of multiple schedulers - section 3.2
3. Real-time producer/consumer paradigm [YARTOS] - section 3.3
4. Coexistence of real-time and conventional OSs - section 3.4
5. Reactive objects and the synchrony hypothesis [synchronous language ESTEREL] - section 3.5

Although much research has been devoted to the design of real-time operating systems, the majority of multimedia processing is performed on the commercial desktop systems running conventional OS. However, when introducing new concepts into commercial products, it is necessary to consider the cost of the development as well as the current investment of the computer community in the conventional systems such as Windows or OS/2. This implies the need for the implementation of cost efficient changes to the current commercial products.

3.1 Predictable operating system protocol

Models and paradigms have been mostly initiated at the Carnegie Mellon School of Computing with MACH 3.0 OS. Multimedia applications require timely service to support time-constrained data types such as digital audio and video. The objectives of such systems include a predictable and reliable distributed real-time environment and a set of tools to allow a system designer to analyse the runtime behaviour at the design stage (Mercer (1994)). Conventional OS's typically employ some time-sharing scheduling algorithms and/ or simple fixed priority scheduling which do not guarantee timely execution of all processes competing for resources. With preemptive kernels an effect known as priority inversion may occur. In summary, scheduling requirements for time-critical threads and conventional threads are incompatible.

The processor capacity reservation mechanism implemented in MACH (Mercer (1993)) takes scheduling of time critical jobs a step further. It allows processes to reserve the capacity they need to run. The principle is based on the separation of the mechanism which controls the resource assumption and delivers resource capacity to programs from an actual policy on how the resources are allocated. Primitives of this protocol use the *reserve abstraction*, *admission policy*, and *reservation enforcement* mechanism to manage CPU allocation. *Reserve* is a kernel entity which represents access to the processor capacity. A reserve contains information about computation time and a reservation period. The reservation request is subject to the system's acceptance or denial. If the request is denied then the process has to decide how to proceed further. The *enforcement*

mechanism measures the usage of each program and uses the associated reservation information to make sure that the usage does not exceed the reservation quantum during any reservation period.

3.2 Coexistence of multiple schedulers

An interesting solution using multiple scheduling policies has been presented in Golub (1994) where the scheduler for the Mach 3.0 microkernel allows multiple scheduling policies to run on the same processor. The conventional *priority* value contains two components: policy and ordering. Within each scheduling policy, a different type of ordering is imposed. Each policy maintains its own *run_queue*. A scheduling policy is modelled as an object with a public interface and private data structures and methods. The selection module scans the highest priority scheduling policy for runnable threads to select the thread to run. The preemption is implemented throughout the clock interrupt and associated routines which determine whether the current thread should be preempted.

Two other implementations similar to the above framework are Solaris 2.1 and OS/2. In Solaris 2.1 fixed priority real-time threads may preempt timesharing threads. The real-time threads are also exempt from priority adjustment rules. In OS/2, the threads are assigned to one of four ordered scheduling classes: real-time, fixed priority, timesharing, and background. Each scheduling class contains 32 priority levels, and time critical threads run at the highest priority class. Although this approach improves the predictability of processing it does not guarantee QoS parameters.

3.3 Real-time producer/consumer

The real-time producer/consumer paradigm advocated by Jeffay (1993) models process interaction as a producer/consumer system with a timing constraint on the rate at which the consumer must service the producer. Essentially, the consumer must process data at the rate they are produced. The semantic basis of the system is based on the client/server paradigm and message passing. Each process has a single input port and multiple output ports. Processes exchange messages through communication channels. A process accepts a message on its input port and emits messages via its output ports. The non-blocking emission of messages is achieved through the assumption that messages are produced with a predefined transmission rate function. A process may be blocked on an *accept* statement but never on an *emit* statement.

YARTOS is a micro-kernel application which supports three basic abstractions: tasks, resources, and messages (Jeffay (1991)). In addition, it allows applications to specify the real-time rate of execution of a task (thread of control) that must be completed within a unit of time. The scheduling model is composed of two abstractions: tasks and resources (for more details see Jeffay (1991)). The task is invoked at sporadic intervals and completes its execution at or before its deadline. The task may require access to shared resources in the mutual exclusion fashion. Tasks are reactive in the sense that they execute only in response to events. An experiment with a videoconferencing system which uses YARTOS and a set of IBM-Intel ActionMedia 750 adapters to support live video and audio has shown very promising results.

3.4 Coexistence of real-time and conventional operating systems

The approach taken in Bollella (1995) is based on the notion of two virtual machines which operate over the shared resources (CPU and other system resources). One machine runs unmodified conventional purpose *OS* and the other runs real-time kernel. The access of both virtual machines to the hardware components is multiplexed. From conceptual point of view, this approach provides

good processing model. However, it seems that there is still some work to be done to complete the model, in particular in the areas of sharing of pointing devices between two virtual machines and in the maintenance of consistency of data structures for shared devices. The overhead of switching between virtual processors seems to be also too high.

3.5 Class of reactive objects

Reactive systems and synchronous programming languages have been discussed in Harel (1985) and Berry (1988). Reactive systems are repeatedly prompted by the outside world to which they continuously respond in the form of emitted outputs. Reactive objects are effectively real-time controllers which accept events from an outside environment and instantaneously generate new events in response (Papathomas (1995)). Reactive objects communicate with each other and other objects in the system through invocations. This concept offers some useful features to enhance the computational model. For example there is no need to introduce a special representation of time since they may accept an external clock event derived from an external clock.

The computational model is then the integrated coexistence of active objects (Gibbs (1991)) which operate under the control of a set of reactive objects - synchronisation managers. Two concepts are employed to implement the model: the concept of abstract states, and the concept of the state notifications. The state notifications allow the other objects to monitor the state changes and to synchronise accordingly. The claim is that the abstract states may be used in synchronisation conditions together with class inheritance thus avoiding or minimising the effect of the *inheritance anomaly* (Satoshi (1993)). The model is designed to handle multiple media streams in real time, manage compatible *QoS* for related streams, and monitor temporal relationships of multiple streams.

4 TECHNOLOGIES FOR THE FUTURE OFFICE

The office of the future will be characterised by the need for real-time processing of highly distributed cooperative and computer-participative multimedia applications. At present, the commercial multimedia systems like Video for Windows, QuickTime, or Video In from IBM generally support efficient storage, retrieval, and presentation of pre-recorded video clips. However, they do not support adequately the direct processing of live media, live orchestrated multimedia presentations with random user interaction, and computer-participative systems.

The reserve abstraction (section 3.1) has a number of advantages. First, the thread receives feedback and may decide on further processing policy. Second, it is possible to build a *QoS* manager which would maintain expected *QoS* parameters². The paradigm based on real-time producers and consumers (section 3.3) has already proved that it is possible to anticipate processing rates of continuous media by using non-blocking emissions of messages to meet real-time processing deadlines. Both architectures require significant changes to operating system architectures as well as to the designer's 'state of mind' to accept new concepts of management of Inter-Process Communications (IPC).

The abstraction presented in section 3.2 which is based on the cooperation of multiple scheduling policies provides a framework which could be accommodated by the current operating systems with relatively low cost. Some modifications of this framework are already used commercially in OS/2

² We have omitted the discussion on unbounded priority inversion in micro-kernel systems

and Solaris 2.1. The coexistence of multiple OS's over a single CPU (section 3.4) seems to carry a high processing overhead. The model based on reactive objects (section 3.5) provides a very sound conceptual and programming framework which does not require drastic changes in operating systems. However, a large processing overhead could be incurred for medium and large size applications which exchange many messages.

The management of the allocation of resources to multiple multimedia applications is the crucial point of delivering quality multimedia presentations to a user. In Figure 2, the video conferencing application requesting the processing rate 30 fps will consume a considerable amount of CPU time. While many videoconferencing applications use the frame rate 7-15 fps, the question is whether it is necessary to deliver video data with this precision.

Figure 2 Resource distribution in a multimedia system.

The challenge is to build a system in which multiple resource reservation policies support the management of multimedia presentations according to the user's perception of quality. The following four concepts should be considered in order to lift multimedia processing from its infancy to a mature technology state:

- 1) Support for temporally sensitive data is controlled via a *QoS* manager which maintains required level of services while optimizing resource allocation. The *QoS* manager's performance observations (events and notifications) are visible to the user and programmer via a high level interface such as an intelligent agent;
- 2) Real-time processes and threads are isolated from non-real time activities by the new class of schedulers.
- 3) The distributed system provides predictable response time;
- 4) Object-Oriented model would interact with the *OS* through a well defined interface (a set of APIs) so that the programmer may easily take advantage of some constructs built into the operating system (eg. synchronisation policies, performance notifications).

5 CONCLUSION

To support predictable end-to-end performance and temporal guarantees for low latency applications is difficult task the success of which depends on many factors. It is out of scope of this paper to provide an exhaustive survey of all approaches which aim at providing the suitable operating environment for multimedia application processing. The research seems to be moving towards providing predictable service to multimedia applications throughout the adaptable resource allocation mechanism. The focus is on isolating real-time activities of multimedia applications from unpredictable interference of other processes, and eliminating unbounded priority inversion.

The movement towards adaptive allocation of resources will promote the development of high level model of Quality of Service. The Quality of Service parameters then will reflect the user's desired quality of presentation. The capability to specify *QoS* parameters at the application level as

the system concept will enable to create coherent conceptual framework for flexible management of shared resources.

6 REFERENCES

- Anderson, D.P., Tzou, S., Wahbe, R., Govidan, R., Andrews, M. (1990), "Support for Continuous Media in the DASH System", *Proc. of the 10th International Conference on Distributed Computing Systems*, pp. 54-61.
- Arbab, F., Herman, I., and Reynolds, G.J., (1993), "An Object Oriented Model For Multimedia Programming", *Proc of Eurographics'93*, Blackwell Publishers, Volume 12, No.3, pp. C-101-C-113.
- Berra, P.B., Chen, C.Y.R. Ghafoor, A., Little, T.D.C. (1992), "Issues in Networking and Data Management of Distributed Multimedia Systems", Invited paper, *Symposium on High-Performance Distributed Computing*, Syracuse New York, September 1992.
- Berry, G., Gonthier, (1988) "The ESTEREL Synchronous Programming Language: Design, Semantics, Implementation", *INRIA report No. 842*, INRIA.
- Bollella, G., Jeffay, K., (1995), "Support for Real-Time Computing within General Purpose Operating Systems: Supporting Co-Resident Operating Systems", *Proc. of the IEEE Real-Time Technology and Application Symposium*, Chicago, May, 1995
- Christopher J. Lindblad, (1994) "A Programming System for the Dynamic Manipulation of Temporally Sensitive Data", *Technical Report MIT/LCS/TR-637*, August, 1994, MIT Laboratory for Computer Science.
- Coulson, G., Blair, G.S., (1992), "Meeting the Real-Time Synchronisation Requirements of Multimedia in Open Distributed Processing", *Technical Report MPG-92-45*, Distributed Multimedia Research Group, University of Lancaster.
- Dannenberg, R.B., Rubine, D., (1994), "A Comparison of Streams and Time Advance As Paradigms for Multimedia Systems", *Technical Report CMU-CS-94-124*, School of Computer Science Carnegie Mellon University, Pittsburgh.
- Gibbs, S., Dami, L. and Tsichritzis, D. (1991), "An Object Oriented Framework for Multimedia Composition and Synchronisation", *MULTIMEDIA Systems, Interaction and Applications* (ed. Kjelldahl, L.), 1st. Eurographics Workshop, Stockholm, April 18/19.
- Golub, D. B. (1994), "Operating System Support for Coexistence of Real-Time and Conventional Scheduling", *Technical Report*, CMU-CS-94-212, School of Computer Science, Carnegie Mellon University, Pittsburg.
- Hardman, L., Q., Bulterman, D.C. (1995), "Authoring Support for Durable Interactive Multimedia Presentations", *Technical Report STAR.ps*, CWI, Multimedia Kernel Systems Project.
- Harel, D., Pnueli, A. (1985) "On the Development of Reactive Systems", in *Logics and Models of Concurrent Systems* (ed. K.R. Apt), *NATO ASI Series F*. Vol. 13, pp. 477-498.
- Hoepner, P. (1991), "Presentation Scheduling of Multimedia Objects and Its Impact on Network and Operating System Support", *2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, 11, pp. 12-23.
- Jeffay, K., Stone, D., F. Donelson Smith (1991), "Kernel Support for Live Digital Audio and Video", *Computer Communications*, Vol. 15, No. 6. (July/August 1992), pp. 388-395
- Jeffay, K., Stone, D., Poirier, Daniel, (1991), "YARTOS, Kernel support for efficient, predictable real-time systems", *Real-Time Systems Newsletter*, Vol. 7, No. 4 (Fall), pp. 8-13.

- Jeffay, K., (1993), "The Real-Time Producer/Consumer Paradigm: A paradigm for the construction of efficient, predictable real-time systems", *Proc. of the 1993 ACM/SIGAPP Symposium on Applied Computing*, Indianapolis, pp. 796-804.
- Jeffay, K., Bennett, D., (1995), "A Rate-Based Execution Abstraction for Multimedia Computing", *Proc. of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, Durham, April 1995, pp. 67-78
- Mercer, C.W, Savage, S., Hideyuki Tokuda (1993), "Processor Capacity reserves: An Abstraction for Managing Processor Usage", *Technical Report*, CMU-CS-93-157, School of Computer Science, Carnegie Mellon University, Pittsburg.
- Mercer, C.W, Zelenka, J., Ragunathan Rajkumar (1994), "On Predictable Operating System Protocol Porcessing", *Technical Report*, CMU-CS-94-165, School of Computer Science, Carnegie Mellon University, Pittsburg.
- Papathomas, M., Blair, G.S., Coulson, G., Philippe Robin (1995), "Addressing the real-time synchronisation requirements of multimedia in an object-oriented framework", in *IS&T/SPIE Proceedings Vol 2417, Multimedia Computing and Networking 1995*
- Rossum, G., Jansen, J., K. Sjoerd Mullender (1993), "CMIFed: A Presentation Environment for Portable Hypermedia Documents", *Tecnicl Report CS-R9304*, ISSN 0169-118X, CWI.
- Satoshi Matsuoka, and Akinori Yonezawa (1993), "*Analysis on Inheritance Anomaly in Object-Oriented Concurrent Programming Languages*", in *Research Directions in Concurrent Object-Oriented Programming* (ed. Gull Agha, P. Wegner, and A. Yonezawa), The MIT Press, Cambridge, MA.
- Staehli R., Walpole J., and Maier D., (1994), "Quality of Service Specification for MultimediaPresentations", <http://tec-reports/Tech. Report 94-033>, Department of Computer Science & Engineering, Oregon Graduate Institute of Science & Technology.
- Williams N., Blair, G.S, (1992), "Distributed Multimedia Application Study", *Technical Report* MPG-92-11, Distributed Multimedia Research Group, University of Lancaster.

7 BIOGRAPHY

Jana Dospisil is a Lecturer at Monash University in Melbourne, Australia. She holds MEng from Technical University in Brno , Czech Republic and M(Comp) from Monash University in Melbourne. Prior to moving to Australia, she worked for thirteen years in a number of software development positions in Czech Republic. Her research interests include multimedia and agent based systems.

Dr Elizabeth A. Kendall is a Senior Lecturer in Computer Systems Engineering at the Royal Melbourne Institute of Technology in Melbourne, Australia. She has a MEng from the Massachusetts Institute of Technology and a MEng a PhD from California Institute of Technology. Prior to moving to Australasia, she worked for eleven years in the U.S. aerospace and defence industry. Her research interests are in the agent based systems and advanced applications of object oriented technology.

Tony Polgar is a Project Manager in ISSC Australia in Melbourne. He graduated with MEng degree from the Technical University in Prag, Czech Republic. He has also a M(Comp) from Monash University in Melbourne. Prior to moving to Australia, he worked for twelve years in a number of software development positions in Czech Republic. He is currently involved in research and teaching at Monash University. His research interests are in project management of large complex software applications.