Real-Time Systems Research Group

Department of Computer Science
University of York, UK

Prospectus

October 1993

The Real-Time Systems Research Group undertakes research into all aspects of the design, implementation and analysis of real-time systems. Specifically, the group is addressing: formal and structured methods for development, scheduling theories, reuse, language design, kernel design, communication protocols, distributed and parallel architectures, and program code analysis. The purpose of this document is to provide an overview of the group's activities and contact information.

Real-Time Systems Research Group

Membership (October 1993)

Neil Audlsey
Alan Burns
Rod Chapman
Peter Cornwell
Rob Davis
Qui Jiancheng
Charles McElone
Stuart Michell
Mark Nicholson
Sasikumar Punnekkat
Mike Richardson
Dave Scholefield
Ken Tindell
Andy Wellings
Ben Whittle
Ning Zhang
Sijing Zhang

Contact Address

Dr A. Burns
Real-Time Systems Research Group
Department of Computer Science
University of York
Heslington
York Y01 5DD, UK
email: burns@minster.york.ac.uk
Tel: +44 (0)904 432779 Fax: +44 (0)904 432767

CONTENTS

Real-Time Systems 3
Main Areas of Activity 3
Research Projects and Sponsors 9
Group Members and Research Interests 10
Electronic Publishing of Technical Reports 11
Recent Publications 13

REAL-TIME SYSTEMS

A real-time system is any information processing system that must respond to input stimuli within a finite and specified time interval. The stimuli being either an event at the interface to the system or some internal clock tick that is, at least notionally, coordinated to the passage of time in the system's environment. Real-time systems are found in a wide range of applications areas, from simple domestic appliances to multi-media systems, large process control plant and safety critical avionics systems. In some systems the required response times are measured in millisecond, in others it is seconds or even minutes. Nevertheless they all have deadlines that must be satisfied. In the production of real-time systems, it is insufficient to merely test for compliance (with the timing requirements) once the system is built. A comprehensive and systematic approach to the specification, design, implementation and analysis of these systems is required.

Practitioners working in the field of real-time computing often distinguish between hard, firm and soft real-time systems. *Hard real-time systems* are those where there is an absolute imperative that response times must occur within the specified time intervals. Failure to meet a hard timing requirement could lead to catastrophic failure of the application. *Firm deadlines* are not safety critical but are absolute; if the response is not delivered by the deadline then there is no value in producing it late. The system has failed, but the failure is benign. The third category, *soft real-time systems*, have deadlines by which the response should occur but the system will still function adequately if the occasional soft deadline is missed. In general there is still value in producing a "soft" result after the deadline. Many of the interesting problems in real-time systems research come from applications that contain hard, firm and soft components.

Although most real-time systems are also embedded, the hardware platforms on which they execute vary considerable. Many current systems still use programmable controllers; single processor systems are also common. There is, however, an increasing use of multiprocessors and full distributed systems. The later introduces important topics such as partial failure (and hence the potential for timely recovery) and time synchronisation (because there is usual no single source of clock information).

MAIN AREAS OF ACTIVITY

The Real-Time Systems Research Group at York is probably one of the largest academic research groups in the world focusing on the engineering of real-time systems. The work of the group spans a wide range of topics, a selection of which is discussed below (scheduling analysis, architectures, HRT-HOOD, TAM, safety kernels and industrial case studies). Work is also been undertaken on symbolic execution (for worst case timing analysis), best effort scheduling, specification techniques for timing requirements, language assessment (in particular Ada9X), communication protocols, real-time databases, reuse and optimisation algorithms (simulated annealing, stocastic evolution and genetic algorithms) for system configuration. The group also benefits from close association with two other research activities at York: the High Integrity Systems Engineering Group and the Formal Systems Research Group.

The aim of the group is to undertake fundamental research, and to bring into engineering practice modern techniques, methods and tool. Areas of application of our work include space and avionic systems, engine controllers and multi-media systems. Work has been funded by SERC and DTI, BAe, European Space Agency (ESA), NASA, Defence Research Agency (DRA), Rolls Royce Aeroengines, and the Health and Safety Executive (HSE).

1. Scheduling Analysis

One of the most important aspects of the work in real-time at York is the development of appropriate mathematical analysis: when any real-time system is engineered it is important that the engineer knows beforehand if deadlines are going to be met. Other branches of engineering, for example civil engineering, or aerodynamics, require mathematical underpinning, and real-time systems are no different. This mathematical analysis has focussed on hard real-time systems.

The work of the Real-Time Systems Research Group is based around the fixed priority preemptive scheduling algorithm: this is a simple scheduling algorithm where a processor executes multiple tasks, each of which is assigned a fixed priority. If a task becomes runnable and has a higher priority than the currently running task then the running task stops (i.e. is preempted) within a bounded time, and the new higher priority task runs. It is important to realise that the priority of a task does not reflect its criticality — priority is merely an artifact of the scheduling algorithm.

There are some difficulties with the multi-tasking approach, since if two tasks were to share certain data there could be the problem of attempting to update the data concurrently, leading to corrupted data. One solution to this is to use a *protected object*: the shared data is hidden inside the object, and a number of methods (i.e. functions and procedures) are available to be called by the tasks. Once a task begins executing a method no other task can execute any method in the protected object. To achieving mutual exclusion, for a protected object, a priority (that is higher than any task that uses the object) can be assigned. As a task enters a protected object its priority is raised to this ceiling level (and hence no other task can gain access). It may seem that there is now the possibility of deadlock, where no task is able to call the method it requires. However, mathematical analysis has been developed that shows it is impossible for deadlock (or livelock) to occur.

One of the problems with developing a task-based approach compared to a simple static cyclic schedule is the problem of jitter: variability on when an output (or input) operation is performed leads to jitter, which can be a problem for many control applications. Our approach can bound jitter to specified levels. For example, on a control system where a sensor is polled every 100ms it may be required that the sensor is read within a 2ms window every 100ms. Our analysis can show how such jitter requirements can be met.

There is often confusion over the use of interrupts in real-time systems: because interrupts can come at arbitrary times there is worry that the effects are unknown. However, an interrupt can be treated mathematically as a high priority task that simply pre-empts any other running task (indeed, the operating system could be designed to make interrupts look like tasks). As long as any data shared between an interrupt handler and a task is placed inside a protected object, there is no problem. The mathematical analysis guarantees that as long as there is some minimum time between successive interrupts from the same source then the effects of the interrupt can be bounded. Indeed this analysis can be extended to predict the behaviour of any sporadic activity.

Analysis has also been produced that will deal with tasks with variable computation times and periods, tasks with deadlines greater than their periods, and tasks with deadlines prior to completion. For these conditions optimal priority ordering algorithms have been developed. The analysis has also been extended to deal with communication protocols (on a LAN) and even disk scheduling for multi-media applications.

The worst-case time a task takes to execute a given piece of code needs to be determined before the mathematical analysis can be used. This time is known as the worst-case execution time (WCET). Finding this time using testing and measuring approaches is not reliable because of the difficulty in finding the longest path through a piece of code, especially when loops are permitted. At York we have developed techniques to find the WCET of a piece of code; we have focused on complex processor architectures including caching and pipelining. We have also

developed a tool that will determine the WCET of Ada code when compiled with the York Ada compiler and targeted to a 68020 VME board.

The mathematical analysis that underpins all of the real-time engineering approach is starting to be embodied in software engineering tools. These tools can play a part early on in the engineering life-cycle, where an early indication of a real-time problem is needed. So, for example, it might be spotted early on that a particular subsystem requires too much computation time, and that it will have to be reduced if other subsystems are to fit. They can also play a part during the coding and testing phases where the sources of potential timing faults can be indicated.

2. The DrTEE Architecture

The DrTEE project is concerned with Distributed Real-Time Execution Environments (hence the acronym DrTEE). Building upon the key results in hard real-time system feasibility analysis developed by the group, a paper architecture for the implementation of hard real-time systems has been developed. The architecture is based upon priority pre-emptive scheduling of periodic and sporadic processes, with those processes able to share resources (protected by mutual exclusive access) and communicate across a LAN using a bounded communications protocol.

The development of the paper DrTEE architecture has required the solution of many outstanding issues in hard real-time system analysis. Whilst this analysis can be proved in a mathematical sense, greater weight is lent to our analysis and architecture if they can be proved using an actual hardware testbed. Therefore, the immediate aim of the DrTEE project is the provision of a testbed to enable experiments in kernel design, architecture design and to check our feasibility analysis.

The testbed takes the form of a number of multiprocessor nodes connected by a LAN. Currently, each node consists of a number of PCs. One PC in each node is dedicated to be an "intelligent" communications processor, linked to other such PCs in other nodes by ethernet. The other PCs in each node are dedicated to the execution of application processes. The PCs within a node are also connected via ethernet (although genuine shared memory within a node is planned for the future). Software protocols giving bounded behaviours are run on top of the ethernet protocols.

The testbed has required the design of a predictable and bounded kernel to provide a basis upon which application processes can be executed. The kernel has been written for a bare PC, using only the BIOS for booting the kernel, thereafter not using BIOS, DOS or other PC software. This has enabled the kernel to be predictable, reliable and bounded; providing an adequate platform for the development of safety-critical applications.

3. HRT-HOOD — A Structured Development Method

Most structured design methods claim to address the needs of hard real-time systems. However, few contain abstractions which directly relate to common hard real-time activities, such as periodic or sporadic processes. Furthermore, the methods do not constrain the designer to produce systems which can be analysed for their timing properties.

Any design method must have an underlying computational model. The key objective is to have a model which facilitates both formal development and schedulability analysis, and yet allows a realistic set of application requirements to be addresses. The following brief description of the computational model for HRT-HOOD exactly matches that described earlier for undertaking effective schedulability analysis.

The task is chosen as the basic computational component, where a task represents a single thread of control. Tasks are pre-emptively scheduled at run-time (as described in section 1). Tasks fall into two basic categories: periodic and aperiodic/sporadic. Periodic tasks typically

sample data or execute a control loop on a regular basis, and have explicit deadlines that must be met. Aperiodic, or sporadic, tasks usually arise from asynchronous events outside the embedded computer. These tasks have specified response times associated with them. To allow worst case calculations to be made, there is a defined minimum time interval between any two aperiodic events (from the same source) or a maximum arrival frequency. If this is the case, the task involved is said to be *sporadic*.

Tasks are assigned a priority and a criticality, and at run-time pre-emptive priority based dispatching is assumed. Priorities may be assigned according to the rate or deadline monotonic scheduling schemes or via some other technique. Criticality is assigned according to the importance of the task to the overall safety and functionality of the system. The relationship between priority and criticality is crucial to the effective scheduling of the system.

Where several tasks co-operate with one another, there is a need for communication. The computational model supports data oriented asynchronous communication via the notion of a protected object. A protected object encapsulates shared data and allows controlled access to it via operations which ensure the integrity of the data. Each protected object is also assigned a priority; this is termed the ceiling priority and its value is at least as great as the highest priority of any tasks that calls its operations. The advantage of an asynchronous model of communication is that it facilities the timing analysis of the final system.

This computational model has been the focus of two studies, one has developed a formal model, TAM. The other has concentrated on the development of a structured design method called HRT-HOOD (Hard Real-Time Hierarchical Object Oriented Design). HRT-HOOD is an extension of HOOD, and includes object types which enable common hard real-time abstractions to be represented. The method is described in the context of a hard real-time system life cycle, which enables issues of timeliness and dependability to be addressed much earlier on in the development process. We believe that this will enable dependable real-time systems to be engineered in a more cost effective manner than the current practise, which in effect treats these topics as performance issues. With the HRT-HOOD method strict rules of decomposition and usage force a design to conform to the computational model and to be effectively analysed (using the techniques outlines in section 1). Industrial case studies have been undertaken using HRT-HOOD and initial evaluations indicate that the method is successful.

4. TAM — A Formal Development Method

In most formal development methods there are at least two languages involved, one for the specification task, and one for the design task (often the translation to implementation is ignored, or considered to be trivial). However, an inherent problem with such a ‘multi language’ approach is the lack of method by which suitable designs are arrived at. A combination of experience and guess-work must be used in order to formulate a design, and then verification -- a time consuming task -- is undertaken. If the verification fails then the design task is undertaken again. This cycle is undergone repeatedly until verification is achieved.

To overcome this problem we have developed the ‘Temporal Agent Model’ (TAM) which is a theory centered around a wide-spectrum language in which both specifications and executable programs can be intermixed. A real-time functional specification in TAM is transformed step-by-step into a mixed program containing both specification fragments and executable code. Such transformations continue until a completely executable program is produced which is guaranteed correct with respect to the original specification. The program may then be analysed by run-time schedulability and allocation tools and then executed.

5. Safety Kernels

Computer software is increasingly being used in the design and construction of safety critical systems. Often an application consists of sequential code which executes in a black box environment where it can be isolated from other applications. To improve fault tolerance, reduce cost and increase flexibility, the next generation of safety critical systems may well be structured as a multitasking distributed hard real-time system.

To obtain the same degree of isolation provided in current systems, requires application tasks to be protected from each other. In traditional time sharing systems, it has been the role of the operating system kernel to provide an environment in which application tasks can execute free from interference from other application tasks; the operating system is often said to support "firewalls" (or damage confinement). Multi-task programs are inherently more non-deterministic than sequential programs. Therefore, it is necessary for the operating system kernel to also ensure that the application tasks execute the correct sequences of operations at all times.

In this work we consider the role of an "operating system" kernel in the provision of a trusted computing base on which to build safety critical real-time applications. We assume the role of the kernel is to:

1) provide safety-related mechanisms — a safety-related mechanism is any mechanism whose use by one application task may affect another application task which is crucial to the overall operation of the system; typically safety related mechanisms are associated with shared resources (e.g. the processor, the memory, the network and software resources)
2) police safety related domain invariants — a safety related domain invariant is an application level invariant which must be satisfied if the system is to remain safe; at one extreme this may simply be prohibiting an operation which will violate the system safety invariant, at the other extreme this may involve monitoring timed sequences of kernel operations and where appropriate invoking application level recovery routines where those sequences are violated.

To aid the structuring of the kernel design, we separate the above two components. The first we call the *safe nucleus* (or *safe run-time support system*); it provides the basic computational resources of the systems. The second we term the *safety services*. Together they provide a *safety kernel*. Initial work on this project involves the formal specification, in Z, of the safe nucleus, and an investigation into the appropriate formalisms for specifying the safety invariants and thus the safety services.

6. The ESA Case Study

An important aspect of the work at York is the undertaking of industrial relevant case studies. One has recently been completed for the European Space Agency.

The next generation of on-board computer systems for space applications must meet high standards of reliability, predictability and safety. This is true not only for those spacecrafts with life support systems and real-time embedded control applications, but also for future deep space missions with the requirement for a high degree of autonomy. Historically, on-board applications have used simple scheduling techniques based on cyclic schedulers. These, unfortunately, can require a considerable amount of effort to ensure that the scheduling load is balanced — an apparently simple programming change can result in an inordinate amount of readjustment of minor and major schedules. Furthermore, it is far from clear that such techniques can easily be scaled to meet the demands of the next generation of systems.

In January 1991, the European Space Agency commissioned a study into the practicality of using task-based scheduling techniques in an on-board application environment. The Study was performed by the Real-Time Research Group at York, British Aerospace Space Systems Ltd and York Software Engineering Ltd (YSE). The Case Study made use of (and helped develop):

(1) The mathematical analysis developed for fixed priority scheduling

(2) HRT-HOOD

(3) A sub-set of Ada9X

(4) Worst Case Execution Time tools

(5) Schedulability analysis and simulation tools

The AOCS (Attitude and Orbital Control System) subsystem of the Olympus satellite exists to acquire and maintain the desired spacecraft position and orientation. The AOCS software may operate in six modes of operation, of these the Normal Mode is the most complex and is used for the greatest percentage of the satellites lifetime. It is this mode that the study modelled.

The Normal Mode software is embedded in the Spacecraft Microcomputer Module (SMM) and communicates with the following devices over a serial data bus

- A Telemetry & Telecommand Subsystem (TMTC),
- An InfraRed Earth Sensor (IRES),
- A Digital Sun Sensor (DSS),
- A Rate Gyro Sensor (RGS),
- Four reaction wheels (RWs),
- Thrusters.

Serial bus messages are placed on the bus according to the priority of the transmitting device. The bus has time slots reserved for replies, ensuring that SMM requests for sensor data receive responses within a 960 μs time slot. There is a 10ms real-time clock.

The case study consisted of 3155 lines of Ada source code, corresponding to 62 HRT-HOOD objects. The object code sizes are (in bytes): 116,312 (code), 2,976 (initialised data), and 6,948 (uninitialised data). Note that all these figures are unoptimised; and that the size of memory available for this class of space applications is still limited by technology factors to the order of 128Kbytes.

Comparative testing with the original Olympus equipment yielded the following conclusions:

1) there was no significant different in the quality of control of the satellite (on the system evulator),

2) the two systems were judged to be equivalent in their responses and noise magnitudes.

The study has successfully shown that it is feasible and practical to use a task-based solution to on-board software systems, and to achieve the same level of control as that using the cyclic executive approach.

The study highlighted many positive benefits that came from the adopted approach. These include:

- a sound engineering approach with a mathematical basis.
- a mathematical proof of schedulability resulting in a reduction in testing time; testing can be heavily targeted at logic checking with reduced emphasis on checking worst case loading situations.
- flexible run-time scheduling allows changes to be made in the application structure without the costly redevelopment of cyclic schedules.
- tool support enables the techniques to be used by engineers rather than academic theoreticians

The tools and methods developed in this study are now being promoted and diffused in other activities funded by the European Space Agency. In particular, reference environments centred on

these tools and methods are being developed for the support of Ada embedded software targeting space qualified hardware platforms.

Research Projects and Sponsors

Project	Source of Funds	Collaborators	Duration	Full-Time Staff
Safety Kernels	HSE	YSE	July 93-Jan 95	
Automatic Generation of Cyclic Executives	Rolls Royce		May 93-Oct 93	1
PROTEUS: Understanding Changing Requirements	SERC	U of Newcastle and Loughborough	Jan 93-Jan 94	2
Real-Time Systems Engineering	SERC	FSRG	Aug 89-Sept 95	3
Supporting Ada in a Distributed Hard Real-Time Environment	DRA		Jan 92-Dec 94	1
Hard Real-Time Operating System Kernel	ESA	BAe	Jan 91-Nov 93	
SPIRITS — Supporting Predictable Implementations of Requirements on Timing and Safety	SERC(DTI)	BAe and Admiral	Jan 91-Dec 93	2
Program Code Analysis	BAe	DCSC	Jan 91-Dec 94+	1
Reuse	Rolls Royce	UTC	Oct 93-Sept 98	1

Total research income since 1990 is in excess of one million pounds.

Abbreviations

YSE — York Software Engineering
HSE — Health and Safety Executive
DRA — Defence Research Agency
BAe — British Aerospace
ESA — European Space Agency
SERC — Science and Engineering Research Council
DTI — Department of Trade and Industry
DCSC — Dependable Computer Systems Centre
UTC — University Technology Centre
FSRG— Formal Systems Research Group (at York)

Group Members and Research Interests

In the following table all phone numbers are +44 (0)904 43 X. And all email names are X@minster.york.ac.uk

Name	Position	Phone	Email
	Activities and Interests		
Neil Audlsey	RA	2735	neil
	Real-time systems: feasibility analysis and dynamic use of resources; safety-critical kernel design and implementation.		
Alan Burns	Reader	2779	burns
	Scheduling theory, Real-time languages, architectures including kernels and communication protocols, real-time databases and real-time systems education.		
Rod Chapman	RA	2786	rod
	Programming environment for safety critical hard real-time systems; worst case execution time analysis for SPARK Ada; secure kernels		
Peter Cornwell	RA	2764	cornwell
	Real-time design reuse Object-oriented design for real-time systems		
Rob Davis	RA	2735	robd
	Scheduling theory, specifically the provision of spare capacity and its use in improving system utility. Real-time Kernels.		
Qiu Jiancheng	Visitor	3244	qiu
	Worst case execution time analysis, real-time architectures, support tools.		
Charles McElone	RS (PT)		charlie
	Run-time scheduling, mutiprocessor systems, transputer networks.		
Stuart Michell	RS	2764	stuart
	Concurrent extensions to object- oriented languages with emphasis on the concurrency model and how required synchronisation constraints affect inheritance.		
Mark Nicholson	RA	3243	mark
	optimisation techniques applicable to real-time safety critical systems, safety design emergence and synthesis, integration of analysis and assessment.		

Sasikumar Punnekkat	RS	3241	sasi
	Hard real-Time systems and fault-tolerant computing.		
Mike Richardson	RA	3243	mike
	Architectures and kernel for real-time systems, scheduling theory, optimisation techniques, real-time databases.		
Dave Scholefield	RA	3377	djs
	Formal development methods, refinement.		
Ken Tindell	RA	3377	ken
	Scheduling theory, real-time communications, real-time operating systems, real-time aspects of multi-media.		
Andy Wellings	SL	2742	andy
	Real-time languages and operating systems, safety kernels, architectures, scheduling theory, and real-time reuse.		
Ben Whittle	RA	3375	ben
	Reuse in real-time systems		
Ning Zhang	RA	3243	ning
	Real-time program temporal analysis and tool development, performance studies of distributed real-time systems, simulation and modeling, and communication networks.		
Sijing Zhang	RS	3378	zsj
	Real-Time communications, LANs, communication protocols.		

Electronic Publishing of Technical Reports

We believe it is important for research output to be communicated as quickly as possible to other workers in the field. To this end we place all our research reports in an electronic archive from which copies of papers and reports can be taken.

The archive can be accessed via anonymous FTP. The address of the site is:

```
minster.york.ac.uk
```

(if the name is not recognised then a numeric name can be used: `144.32.128.41`)

The reports and papers are stored in compressed postscript. On a Unix system the postscript can be de-compressed with the compress command:

```
$ compress -d paper.ps.Z
```

The reports and papers are located in the directory:

```
/pub/realtime/papers
```

Also, the file:

```
INDEX
```

contains a list of the available reports and papers. The file:

```
ABSTRACTS
```

gives the abstracts of the listed reports and papers. The source code to some real time tools is also

available, in the directory:

```
/pub/realtime/programs
```

A typical FTP session would look like this:

```
Script started on Wed Oct 27 14:33:34 1993
$ ftp minster.york.ac.uk
Connected to minster.york.ac.uk.
220 minster.york.ac.uk FTP server (York Tue Aug 17 15:42:31 BST 1993).
Name (minster.york.ac.uk:ken): anonymous
331 Guest login ok, send email address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /pub/realtime/papers
250 CWD command successful.
ftp> get INDEX
200 PORT command successful.
150 Opening BINARY mode data connection for INDEX (8862 bytes).
226 Transfer complete.
8862 bytes received in 0.92 seconds (9.4 Kbytes/s)
ftp> quit
221 Goodbye.
$
```

It is important that the compressed files are obtained with binary mode. The command `binary` within the `ftp` program does this.

REAL-TIME SYSTEMS RESEARCH GROUP

PUBLICATIONS (Since 1990)

1. N. Zhang, A. Burns and M. Nicholson, ‘‘Pipelined Processors and Worst Case Execution Time’’, *Real-Time Systems* **5**(4), pp. 319-343 (1993).
2. N. Audsley, A. Burns, M. Richardson, K. Tindell and A. Wellings, ‘‘Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling’’, *Software Engineering Journal* **8**(5), pp. 284-292 (September 1993).
3. A. Burns and A. Wellings, ‘‘Implementing Analysable Hard Real-Time Sporadic Tasks in Ada9X’’, YCS 209, Department of Computer Science, University of York (1993).
4. D. Scholefield and H. Zedan, ‘‘A Standard for Finite TAM’’, YCS 207, Department of Computer Science, University of York (1993).
5. K.W. Tindell, ‘‘Holistic Schedulability Analysis for Distributed Hard Real-Time Systems’’, YCS 197, Department of Computer Science, University of York (1993).
6. A. Burns and A. J. Wellings, ‘‘HRT-HOOD A Structured Design Method for Hard Real-Time Systems’’, YCS 199, Department of Computer Science, University of York (1993).
7. K.W. Tindell and A. Burns, ‘‘Scheduling Hard Real-Time Disk Traffic’’, YCS 204, Department of Computer Science, University of York (1993).
8. N. C. Audsley, A. Burns, M. F. Richardson and A. J. Wellings, ‘‘Data Consistency in Hard Real-Time Systems’’, YCS 203, Department of Computer Science, University of York (June 1993).
9. R.I. Davis, K.W. Tindell and A. Burns, ‘‘Scheduling Slack Time in Fixed Priority Pre-emptive Systems’’, *Proceedings Real-Time Systems Symposium* (December 1993).
10. A. Burns, A.J. Wellings, C.M. Bailey and E. Fyfe, ‘‘The Olympus Attitude and Orbital Control System: A Case Study in Hard Real-time System Design and Implementation’’, pp. 19-35 in *Ada sans frontieres Proceedings of the 12th Ada-Europe Conference, Lecture Notes in Computer Science*, Springer-Verlag (1993).
11. A. Burns, A.J. Wellings and A.D. Hutcheon, ‘‘The Impact of an Ada Run-time System’s Performance Characteristics on Scheduling Models’’, pp. 240-248 in *Ada sans frontieres Proceedings of the 12th Ada-Europe Conference, Lecture Notes in Computer Science*, Springer-Verlag (1993).
12. A. Burns and A.J. Wellings, ‘‘Measuring, Monitoring and Enforcing CPU Execution Time Usage ’’, *Proceedings of the 6th International Workshop on Real Time Ada Issues, Ada Letters* **XIII**(2), pp. 54-64, Also appears in Ada User 13(2) June 1992 pp 73-78 (March/April 1993).
13. A. Burns and A.J. Wellings(Eds), ‘‘Proceedings of the 6th International Workshop on Real Time Ada Issues’’, *Ada Letters* **XIII**(2) (March/April 1993).
14. N. C. Audsley, A. Burns, M. F. Richardson and A. J. Wellings , ‘‘Incorporating Unbounded Algorithms Into Predictable Real-Time Systems’’, *Computer Systems Science and Engineering* **8**(2), pp. 80-89 (April 1993).
15. N. C. Audsley, A. Burns, M. F. Richardson, D.J Scholefield, A. J. Wellings and H.S.M. Zedan, ‘‘Bridging the Gap between Formal Methods and Scheduling Theory’’, YCS.195, Department of Computer Science, University of York (March 1993).
16. A. Burns and A.J. Wellings, ‘‘Bridging the Real-time Gap between Ada 83 and Ada 9X’’, pp. 71-86 in *Ada Year Book*, ed. C. Loftus, IOS Press (1993).

17. A. Burns and A. Wellings, ‘‘Large, Distributed Parallel Architecures for Real-Time Systems’’, pp. 11-20 in *Invited Paper for the IDA/NASA Workshop on Large Real-Time Systems, Washington*, ed. N. Howes (March 1993).

18. A. Burns and A.J. Wellings, ‘‘Dual Priority Assignment: A Practical Method of Increasing Processor Utilisation’’, *Proceedings of the Fifth Euromicro Workshop on Real-time Systems, IEEE Computer Society Press*, Oulu, Finland, pp. 48-53 (June, 1993).

19. N.A. Audsley, K. Tindell and A. Burns, ‘‘The End of the Line for Static Cyclic Scheduling?’’, *Proceedings of the Fifth Euromicro Workshop on Real-time Systems, IEEE Computer Society Press*, Oulu, Finland, pp. 36-41 (June, 1993).

20. A. Burns, A.J. Wellings, C.M. Bailey and E. Fyfe, ‘‘The Olympus Attitude and Orbital Control System: A Case Study in Hard Real-time System Design and Implementation’’, YCS 190, Department of Computer Science, University of York (1993).

21. A. Burns, M. Nicholson, K. Tindell and N. Zhang, ‘‘Allocating and Scheduling Hard Real-Time Tasks on a Point-To-Point Distributed System’’, First IEEE Workshop on Parallel and Distributed Real-Time systems (PDRTS), Newport Beach, California (Apr. 13-16 1993).

22. N.C. Audsley, A. Burns and A.J. Wellings, ‘‘Deadline Monotonic Scheduling Theory and Application’’, *Control Engineering Practice* **1**(1), pp. 71-78 (1993).

23. A. Burns and A.J. Wellings, *Hard Real-time HOOD: A Design Method for Hard Real-time Ada 9X Systems*, Towards Ada 9X, Proceedings of 1991 Ada UK International Conference, IOS Press (1992).

24. K. Tindell, ‘‘An Extendible Approach for Analysing Fixed Priority Hard Real-Time Tasks’’, YCS 189, Department of Computer Science, University of York (December 1992).

25. K. Tindell, ‘‘Using Offset Information to Analyse Static Priority Pre-Emptively Scheduled Task Sets’’, YCS 182, Dept of Computer Science, University of York (August 1992).

26. N. C. Audsley, A. Burns and A.J. Wellings, *Unbounded Algorithms, Predictable Real-Time Systems and Ada9X,*, IEEE Workshop on Imprecise and Approximate Computation, Phoenix, Arizona (1992).

27. K. Tindell, A. Burns and A.J. Wellings, ‘‘Allocating Real-Time Tasks (An NP-Hard Problem made Easy)’’, *Real-Time Systems* **4**(2), pp. 145-165 (June 1992).

28. A. Burns(Ed.), *Towards Ada 9X*, Proceedings of 1991 Ada UK International Conference, IOS Press (1992).

29. N. C. Audsley, A. Burns, M. F. Richardson and A. J. Wellings, ‘‘Incorporating Unbounded Algorithms Into Predictable Real-Time Systems’’, YCS 171, Department of Computer Science, University of York (March 1992).

30. A. Burns and A.J. Wellings, ‘‘In Support of the Ada 9X Real-Time Facilities’’, *Ada Letters* **XII**(1), pp. 53-64 (Jan/Feb 1992).

31. A. Burns and A.J. Wellings, ‘‘Designing Hard Real-time Systems’’, pp. 116-127 in *Ada: Moving Towards 2000, Proceeedings of the 11th Ada-Europe Conference, Lecture Notes in Computer Science Vol 603*, Springer-Verlag (1992).

32. N. C. Audsley, A. Burns, M. F. Richardson and A. J. Wellings, ‘‘Hard Real-Time Scheduling: The Deadline Monotonic Approach’’, pp. 127-132 in *Real-Time Programming*, ed. W. A. Halang and K. Ramamritham, Pergamon Press (1992).

33. N. C. Audsley, A. Burns, M. F. Richardson and A. J. Wellings, ‘‘Absolute and Relative Temporal Constraints in Hard Real-Time Database’’, *Proceedings of the Fourth Euromicro Workshop on Real-time Systems, IEEE Computer Society Press*, Athens, pp. 148-153 (June

1992).

34. N.C. Audsley, A. Burns, M.F. Richardson and A.J. Wellings, ‘‘Deadline Monotonic Scheduling Theory’’, pp. 55-60 in *Proceedings IFAC/IFIP International Workshop on Real-time Programming, WRTP’92, Bruges* (June 1992).
35. N. C. Audsley, A. Burns, M. F. Richardson and A. J. Wellings, ‘‘Incorporating Unbounded Algorithms into Predictable Real-Time Systems’’, YCS 171, Dept of Computer Science, University of York (March 1992).
36. A. Burns and A.J. Wellings, ‘‘Safety Kernels and the Ada Programming Language’’, pp. 56-70 in *Ada in Transition, Proceedings of 1992 Ada UK International Conference* (1992).
37. K. Tindell, A. Burns and A. Wellings, ‘‘Mode Changes in Priority Pre-emptive Scheduled Systems’’, *Proceedings Real Time Systems Symposium*, Phoenix, Arizona, pp. 100-109 (December 1992).
38. C.M. Bailey, A. Burns, E. Fyfe, F. Gomez-Molinero and A.J. Wellings, ‘‘Implementing Hard Real-time Systems: A Case Study’’, *Proceeeding International Symposium on Real-time Embedded Processing for Space Applications, Les Saintes-Maries-de-la-Mer, France* (November 1992).
39. N. Audsley, A. Burns, M. Richardson, K. Tindell and A. Wellings, ‘‘Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling’’, *Report RTRG/92/120 Department of Computer Science, University of York* (February 1992).
40. A. Burns, ‘‘Scheduling Hard Real-Time Systems: A Review’’, *Software Engineering Journal* **6**(3), pp. 116-128 (1991).
41. A. Burns and A.J. Wellings, ‘‘Priority Inheritance and Message Passing: A Formal Treatment’’, *Real-time Systems* **3**(1), pp. 14-44 (1991).
42. N.C. Audsley, A. Burns, M.F. Richardson and A.J. Wellings, ‘‘Hard Real-Time Scheduling: The Deadline Monotonic Approach’’, *Proceedings 8th IEEE Workshop on Real-Time Operating Systems and Software*, Atlanta, GA, USA (15-17 May 1991).
43. N.C. Audsley, A. Burns, M.F. Richardson and A.J. Wellings, ‘‘STRESS: A Simulator for Hard Real-Time Systems’’, RTRG 106, Department of Computer Science, University of York (1991).
44. A. Burns and A. M. Lister, ‘‘A Framework for Building Dependable Systems’’, *Computer Journal* **34**(2), pp. 173-181 (1991).
45. N. C. Audsley, K. Tindell, A. Burns, M. F. Richardson and A. J. Wellings, ‘‘The DrTee Architecture for Distributed Hard Real-Time Systems’’, *Proceedings 10th IFAC Workshop on Distributed Control Systems*, Semmering, Austria (9-11 September 1991).
46. N. C. Audsley, ‘‘Optimal Priority Assignment and Feasibility of Static Priority Tasks With Arbitrary Start Times’’, YCS 164, Dept. Computer Science, University of York (December 1991).
47. N. C. Audsley, ‘‘Resource Control For Hard Real-Time Systems: A Review’’, YCS 159, Department of Computer Science, University of York (August 1991).
48. A. Burns, A.M. Lister and J.A. McDermid , ‘‘TARDIS: an architectural framework for timely and reliable distributed information systems’’, *Proc. Aust. Software Engin. Conf (ASWEC ’91)*, Sydney, pp. 1-15 (July 1991).
49. A. Burns and A.J. Wellings, ‘‘Criticality and Utility in the Next Generation ’’, *Real-time Systems* **3**(4), pp. 351-354 (1991).
50. N Audsley, A. Burns, M. Richardson and A.J. Wellings, ‘‘A Database Model for Hard Real-Time Systems’’, RTRG 103, Department of Computer Science, University of York

(1991).

51. A. Burns, ‘‘A Framework for Building Real-Time Responsive Systems’’, in *Proceedings of International Workshop on Responsive Computer Systems, Golfe Juan* (October 1991).
52. N. C. Audsley, K. Tindell, A. Burns, M. F. Richardson and A. J. Wellings, ‘‘The DrTee Architecture for Distributed Hard Real-Time Systems’’, *Proceedings IEEE Workshop on Architecture Support for Real-time Systems*, San Antonio, Texas (December 1991).
53. A. Burns and A.J. Wellings, ‘‘Specifying an Ada Tasking Run-time Support System’’, *Ada User* **12**(4), pp. 160-186 (December 1991).
54. A.D. Hutcheon, D.S. Snowden and A.J. Wellings, ‘‘Programming and Debugging Distributed Target Systems’’, pp. 209-251 in *Integrated Project Support Environments: The ASPECT Project*, ed. A.W. Brown (1991).
55. A. Burns and A.J. Wellings, *Real-time Systems and their Programming Languages*, Addison Wesley (1990).
56. A. Burns and A.J. Wellings, ‘‘The Notion of Priority in Real-time Programming Languages’’, *Computer Languages* **15**(3), pp. 153-162 (1990).
57. A. Burns, A.M. Lister and A.J. Wellings, ‘‘Ada and Distributed Systems’’, *Encyclopedia of Computer Science and Technology* **21 (Supplement 6)** (1990 $P 1-18).
58. A. Burns and A.J. Wellings, ‘‘Real-time Ada: Outstanding Problem Areas’’, *Proceedings of the 3nd International Workshop on Real Time Ada Issues, ACM Ada Letters, Ada Letters* **X**(4), pp. 5-14 (1990).
59. A. Burns and A.J. Wellings, ‘‘Usability of the Ada Tasking Model’’, *Proceedings of the 3nd International Workshop on Real Time Ada Issues, ACM Ada Letters, Ada Letters* **X**(4), pp. 49-56 (1990).
60. A. Burns, ‘‘Distributed Hard Real-Time Systems: What Restrictions are Necessary?’’, pp. 297-304 in *Real-time Systems Theory and Applications*, ed. H. Zedan, Elsevier Scientific (1990).
61. A. Burns and T.J. Quiggle, ‘‘Effective Use of Abort in Programming Mode Changes’’, *Ada Letters* (1990).
62. M. Pilling, A. Burns and K. Raymond, ‘‘Formal Specification and Proofs of Inheritance Protocols for Real-Time Scheduling’’, *Software Engineering Journal* **5**(5), pp. 263-279 (1990).
63. N. Audsley and A. Burns, ‘‘Real-Time System Scheduling’’, YCS.134, Department of Computer Science, University of York (1990).
64. A. Burns and M. Richardson, ‘‘A Database Model for Real-time Systems’’, YCS.144, Department of Computer Science, University of York (Aug 1990).
65. A Burns, A.J. Wellings and G.L. Davies, ‘‘Asynchronous Transfer of Control and the Ada Programming Language’’, *Proceedings of the 4th International Workshop on Real Time Ada Issues, Ada Letters* **10**(9) (1990).
66. A. Burns, A.J. Wellings and H. Zedan, ‘‘An Assessment of the use of Occam for Dependable Real-Time Systems’’, pp. 287-294 in *Proc. 13th Occam User Group Technical Meeting%E H. Zedan*, IOS Press (1990).
67. A.D. Hutcheon and A.J. Wellings, ‘‘The York Distributed Ada Project’’, pp. 67-104 in *Distributed Ada: developments and experiences*, ed. J. Bishop, Cambridge University Press (1990).
68. A.D. Hutcheon and A.J. Wellings, ‘‘Tolerating Processor Failure using the Virtual Node Approach to the Distributed Execution of Ada Programs’’, pp. 225-236 in *Real-time*

Systems Theory and Applications, ed. H. Zedan, Elsevier Scientific (1990).

69. A.B. Gargaro, S.J. Goldsack, R.A. Volz and A.J. Wellings, ‘‘Supporting Reliable Distributed Systems in Ada 9X: An Initial Proposal’’, pp. 292-321 in *Distributed Ada: developments and experiences*, ed. J. Bishop, Cambridge University Press (1990).

70. A.B. Gargaro, S.J. Goldsack, R.A. Volz and A.J. Wellings, ‘‘Towards Supporting Distributed Systems in Ada 9X’’, pp. 310-323 in *Ada: Experiences and Prospects, Proceedings Ada-Europe Conference, Dublin*, Cambridge University Press (1990).

71. A.B. Gargaro, S.J. Goldsack, R.K. Power, R.A. Volz and A.J. Wellings, ‘‘Adapting Ada for Distribution and Fault Tolerance’’, *Proceedings of the 4th International Workshop on Real Time Ada Issues, Ada Letters* **10**(9) (1990).

72. K. Tindell, ‘‘Dynamic code replacement and Ada’’, *Ada Letters* **10**(7), pp. 47-54 (Sept.-Oct. 1990).

73. K. Tindell, ‘‘Allocating Real-Time Tasks (An NP-Hard Problem made Easy)’’, YCS 149, Department of Computer Science, University of York (December 1990).

74. N. C. Audsley, ‘‘Deadline Monotonic Scheduling’’, YCS 146, Department of Computer Science, University of York (October 1990).