The Use of Preemptive Priority-Based Scheduling for Space Applications†

C.M. Bailey and E. Fyfe
British Aerospace Space Systems Ltd
Communication Satellites Division
Stevenage
UK

T. Vardanega
ESTEC
European Space Agency
Noordwijk
The Netherlands

A.J. Wellings
Real-time Systems Research Group
Department of Computer Science
University of York
York, UK

Abstract

In January 1991, the European Space Agency commissioned a study into the practicality of using process-based scheduling techniques in an on-board application environment, with Ada as the implementation language. This short paper summarises the results of that study.

1. Introduction

The next generation of on-board computer systems 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. This short paper reports on a study which primarily focused on predictability aspects. 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 process-based scheduling techniques in an on-board application environment. The Study was performed by British Aerospace Space Systems Ltd, The University of York and York Software Engineering Ltd (YSE). It was performed in two consecutive phases. The primary activities of the 1st phase were

1) A review of current scheduling theory, the selection of one theory to be used in the study, and the application of the theory to Ada 83 and Ada 9X.
2) The definition and development of a structured design methodology to support the design of hard real-time systems.
3) The definition of software tools to support the scheduling method, and the definition of any changes required to an Ada compiler and its stand-alone run-time support system.

Phase 2 has involved:

1) the implementation of the identified tools, and changes to the Ada compiler and run-time system
2) the design and implementation of a realistic case study.

The study is now complete, and this short paper summarised its experiences, its results and its conclusions. In section 2 we identify which scheduling theory we adopted and briefly describe the reasons for doing so. We also describe the modification to the Ada Compiler and its run-time support system which were needed to support the approach. In section 3 we provide a short overview of the structured design method we have developed, and in section 4 we summarise the tools produced by the project. In section 5 we describe the case study and give our experiences and results. Finally, in section 6 we present our evaluation and conclusion.

2. The Scheduling Approach and Ada

Before deciding on a scheduling approach, a broad survey of current and future space applications was carried out to identify their typical characteristics[4]. This indicated that:

- the process mix is predominately periodic but with a

† The work has been supported by the European Space Agency (ESTEC Contract 9198/90/NL/SF).

significant aperiodic presence

- aperiodic sources generally represent external sources such as telemetry, telecommands and sensors
- the environment is typically deterministic in the sense that the number and types of objects to be monitored, controlled or analysed is fixed during early design phase
- both hard and soft timing constraints exist.

Given the above characteristics it was decided that preemptive priority based scheduling was the most appropriate technology to use, and that Deadline Monotonic Scheduling Analysis[17] was more suitable than the usual Rate Monotonic Analysis[18] because:

- it allows the deadline of a process to be less than its period
- it allows sporadic processes to be more easily modelled
- it allows for better control over jitter.

Although Ada 83 has been widely criticised for it support for real-time, the limitations are well understood[19, 6, 7] and extensive changes have been made in Ada 9X to make the language more responsive to the needs of the real-time community[16, 1]. The approach adopted by the project has been to use a restricted subset of Ada (e.g., no dynamic tasking, no dynamic memory allocation, no recursion) and to extend the Ada Compiler and its run-time system to support[13]:

- a large priority range (64 levels of software priority)
- passive tasks (with the semantics of Ada 9X protected types)
- a "delay until" function (to allow a more accurate representation of a periodic task)
- monotonic time (to avoid the problems with using a time-of-day clock)
- time fences (for the detection of worst case execution overruns of tasks)

Note that all of the above functionality, except time fences, will be available in Ada 9X.

3. Hard Real-time HOOD

HRT-Hood is a new design methodology that builds on the foundations of HOOD[2]. It combines object oriented design and hierarchical decomposition with explicit abstractions which support common hard real-time design paradigms. It was developed because it was felt that none of the so-called real-time design methods adequately supported the development of hard real-time systems[8].

HRT-HOOD recognises the following object types:

- passive — objects which have no concurrency and no synchronisation constraints (similar to passive objects in HOOD)
- active — general objects modelling concurrent entities (similar to active objects in HOOD)
- cyclic — objects which represent periodic activities
- sporadic — objects which represent aperiodic, or sporadic, activities
- protected — objects which control access to resources
- class — generic objects (similar to those found in HOOD)

Objects are described by their operations, their threads of control, their synchronisation with other objects, and their real-time attributes (e.g. deadline, period, worst case execution time, criticality). For a full description of HRT-HOOD, the reader is referred to the literature[9, 8].

4. Support for the Approach

In order to support the project goals it was necessary to write several tools. The following were constructed:

Worst Case Execution Time: All scheduling analysis techniques require that the execution time of a process/task be known. The tool produces estimates of the time that will be spent in sections of an Ada program, specifically the time between task interactions (or references to passive tasks). Our approach has been to analyse the object code produced by the compiler and the assembler[14]. The York Ada compiler adds the Ada structural data from the source code to the symbolic debugging section of each file of object code. This requires only a few lines of code to be added in a few places, (unlike instrumenting all generated code). A separate program takes the final object file, disassembles it, and breaks the resulting instruction sequence into (machine code) basic blocks. It uses the structural summary from the compiler to assign the time for each basic block to the corresponding high-level construction.

Schedulability Analysis Tool: This tool takes as its input a description of the real-time characteristics of an application and the real-time characteristics of the execution environment (including the overheads incurred by the Ada run-time environment[15]). It performs deadline monotonic scheduling analyses using the test given by Audsley et al[3]. The tool outputs details of the results and allows the user to perform refinement techniques such as period transformations etc.

Schedulability Simulation Tool: This tool allows the run-time scheduling decisions to be simulated in the host environment. It takes as its input a description of the real-time characteristics of the application and the real-time characteristics of the execution environment. Its output includes a trace of the scheduling that would be performed at run-time. The tool can be used in conjunction with the Schedulability Analysis Tool to help the system designer construct a schedulable system.

5. The Olympus Attitude and Orbital Control System

In this section we discuss our experiences with using the above methods, tools and techniques in the redesign and implementation of the Olympus Attitude and Orbital Control System (AOCS).

5.1. The System and its Design

The Olympus satellite was launched in July 1989 as the world's largest and most powerful civil three-axis-stabilised communications satellite. Situated at longitude 19 degrees West, Olympus provides direct broadcast TV and 'distance learning' experiments to Italy and Northern Europe.

The AOCS subsystem 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 elected to model.

Hardware Architecture

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. Gyros have the highest priority, followed by the TMTC and the SMMs. 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.

Software Requirements

The object of 'Normal Mode' attitude control is to maintain the satellite's orientation to Earth. This is to be achieved by a 200 ms cyclic task that forces IRES roll and pitch angles and a rate-gyro derived yaw angle to zero by controlling the speed of four reaction wheels.

For two spells per day of 15 minutes, a one second period task calibrates the gyro drift rate by comparing the yaw estimate, derived from an integration of gyro rate, against the sun and earth positions. The gyro angle and gyro drift rate are corrected at the end of each spell. The gyro data is received approximately every 100ms (without being requested).

In addition to the regular activities identified above, further attitude control functions occur at less frequent or irregular intervals.

- Momentum dumping is triggered when the speed of any reaction wheel exceeds a preset threshold. This consists of a reduction in the reaction wheel speed in a series of steps, while compensating bursts of thruster firings to prevent the loss of Earth-pointing. Dumping on the three axes operates independently.
- The Telemetry and Telecommand Subsystem (TMTC) routinely requests status information from the AOCS software. This task, which has a minimum period of 62.5 ms, keeps the ground informed of the spacecraft state. The SMM is unable to respond to a telemetry request in the same bus time slot, so it transmits a response in a later time slot.
- A telecommand function allows ground to enable or disable control, to enable or disable dumping, to trigger gyro calibration, or to set a reaction wheel failed or operational. Telecommands can occur at a minimum interval of 190 ms.

For a full description of the Requirements see Bailey[5].

The application selected contains many typical features of embedded real-time space software[4], namely:

- Cyclic tasks,
- Sporadic tasks,
- Hard real-time tasks, Soft real-time tasks,
- Background tasks,
- Communication over a bus.

Currently, the operational software is coded in 9989 assembler and scheduled by a cyclic scheduler.

Software Design

The software redesign of the AOCS has been presented elsewhere[10, 11].

5.2. Summary of Results and Experiences

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; note also that the size of memory available for this class of space applications is still limited by technology factors to the order of

128Kbytes.

The result of the analysis using the WCET and Schedulability Analysis tools indicated that all hard real-time tasks were schedulable, but one soft task (implementing telecommands) was not guaranteed to meet its deadline in the worst case. In running the systems, the task did meet its deadline because the analysis was pessimistic in:

1) the analysis techniques — some of the objects have time offsets specified relative to other objects; the equations we were using assumed all tasks had a critical instance when this is clearly not the case.
2) the kernel — in order to take into account the overheads introduced by the kernel, we had to make some assumptions in the analysis techniques; these, on closer inspection, were a little pessimistic[12].
3) the WCET tool — we estimate that our worst case execution time tool[14] is between 5-25% pessimistic because the tool does not model the m68020 internal pipeline, and because some hardware times are data dependent and the tool has to assume the worst case however unlikely it may really be.
4) Calculated execution times assume the worst case paths through the code. In practise these will not occur.

Comparisons with the Original Equipment

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

1) there was no significant different in the quality of control
2) the two systems were judged to be equivalent in their responses and noise magnitudes, except for during Gyro Calibration; although both systems performed to within the precision of the gyro, the case study produced superior results due to its utilisation of floating point maths against the integer maths of Olympus.

Table 1 compares the major parameters of the two systems. It should be noted that neither memory usage nor utilisation were optimised during the study.

	Olympus 1	Case Study
Processor	SBP9989	MC68020
Processor clock	4 MHz	16.67 MHz
Cache	No	Not used
RAM memory	32 KBytes	1 MByte
Word length	16 bits	32 bits
Integer representation	16 & 32 bits	16 & 32 bits
Floating point ´size ´mantissa coprocessor	None	Yes 32 bits 21 bits MC68881
Scheduling method	Cyclic executive	Preemptive priority-based scheduling
No of tasks/functions modelled	32	13
Predominant control-loop arithmetic	Integer	Floating Point
No of HRT-HOOD Objects	not applicable	62
Worst case utilisation	84%	93% (estimated)
Memory used Application Run-time system Total	(Approx) 19 KB 5KB 24KB	(Unoptimised) 98KB 28KB 126KB

Table 1: Comparison between Case Study and Olympus 1

6. Evaluations and Conclusions

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

It is clear that using the ideas and methods developed for the case study do have some adverse cost impact; in particular:

- overheads are incurred by the run-time support system to support Ada tasking
- there is a need to invest in support tools and training to achieve the maximum benefits from approach

However, the study highlighted the fact that the positive benefits far outweigh the disadvantages. These include:

- a sound engineering approach with a mathematical basis
- a mathematical proof of schedulability resulting in a reduction in test duration; 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.

Acknowledgement

The authors would like to acknowledge the contribution of Alan Burns to the ideas presented in the paper, and for his major input into the project.

The authors would also like to thank Paco Gomez Molinero and Fernando Gonzalez-Barcia for their help during the course of the project.

References

1. Ada 9X Mapping/Revision Team, Intermetrics, ‘‘Ada 9X Reference Manual, Draft Version 3.0’’, Ada 9X Project Report (June 1993).
2. European Space Agency, ‘‘HOOD Reference Manual Issue 3.1’’, HRM/91-07/V3.1 (July 1991).
3. N.C. Audsley, A. Burns and A.J. Wellings, ‘‘Deadline Monotonic Scheduling Theory and Application’’, *Control Engineering Practice* **1**(1), pp. 71-78 (1993).
4. C. Bailey, ‘‘Survey of Typical Space Applications’’, Task 6 Deliverable on ESTEC Contract 9198/90/NL/SF, British Aerospace Space Systems Ltd. (September 1991).
5. C. Bailey, ‘‘Software Requirements Document for the Olympus AOCS’’, Task 10 Deliverable on ESTEC Contract 9198/90/NL/SF, British Aerospace Space Systems Ltd. (March 1992).
6. 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).
7. 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).
8. A. Burns and A.J. Wellings, ‘‘Designing Hard Real-time Systems’’, pp. 116-127 in *Ada: Moving Towards 2000, Proceedings of the 11th Ada-Europe Conference, Lecture Notes in Computer Science Vol 603,* Springer-Verlag (1992).
9. 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).
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 688,* Springer-Verlag (1993).
11. 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).
12. 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 688*, Springer-Verlag (1993).
13. 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).
14. C.H. Forsyth, ‘‘Implementation of the Worst-Case Execution Time Analyser’’, Task 8 Volume E, Deliverable on ESTEC Contract 9198/90/NL/SF, York Software Engineering Limited, University of York (June 1992).
15. A.D. Hutcheon, ‘‘Timings of Run-time Operations in Modified York Ada’’, Task 8 Volume C, Deliverable on ESTEC Contract 9198/90/NL/SF, York Software Engineering Limited, University of York (July 1992).
16. Intermetrics, ‘‘Ada9X Integrated Language Specification’’, Version 1.2 (January 1993).
17. J.Y.T. Leung and J. Whitehead, ‘‘On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks’’, *Performance Evaluation (Netherlands)* **2**(4), pp. 237-250 (December 1982).
18. C.L. Liu and J.W. Layland, ‘‘Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment’’, *JACM* **20**(1), pp. 46-61 (1973).
19. L. Sha and J. B. Goodenough, ‘‘Real-Time Scheduling Theory and Ada’’, *IEEE Computer* (April 1990).