POLITÉCNICA

iptc Information Processing and Telecommunications Center

STRAST

The origins of the Open Ravenscar Kernel (ORK)

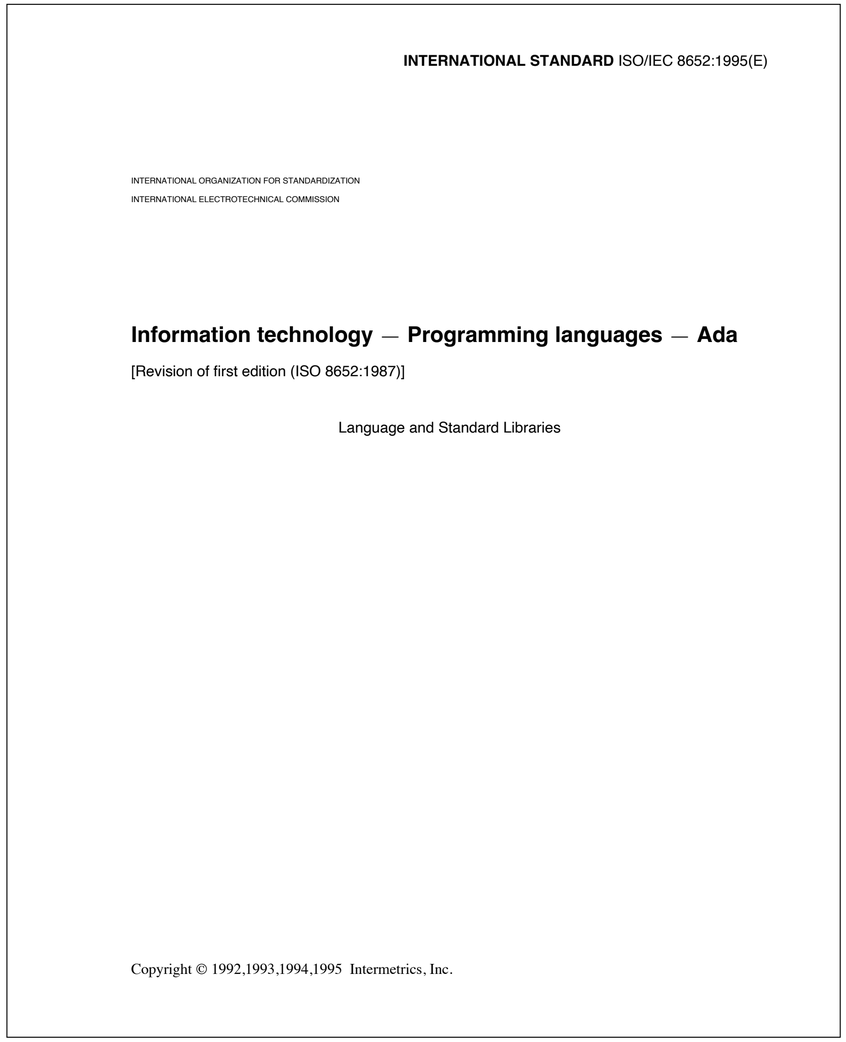
Juan A. de la Puente
Universidad Politécnica de Madrid

Content

- Background: Ada9X & GNAT
- The Laredo 95 Seminar: a bit of inspiration
- The Ravenscar profile
- A first attempt: JTK
- The ORK project
- The project team

Ada 9X & GNAT

- In 1988 the Ada 9X project was launched by the AJPO in order to revise the original ANSI/MIL standard and make it an ISO standard
- The result was Ada 95
  - packages, generics, inheritance
  - protected types, real-time annex
  - first standardized object-oriented language
- At the same time it funded a NYU team to develop an open source compilation system for Ada 9X
- The result was GNAT
  - free, open source
  - very efficient compiler technology



INTERNATIONAL STANDARD ISO/IEC 8652:1995(E)

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION

Information technology — Programming languages — Ada

[Revision of first edition (ISO 8652:1987)]

Language and Standard Libraries

A bit of inspiration

- The Laredo Seminar (1995)

95/07/17-21:

Real-Time Systems Programming: Impact of the new Ada 95 and POSIX Standards

Laredo, Cantabria, Spain
A one week course on Ada 95 and POSIX to be held (in English) by a University in the north of Spain: a service to the international Ada community (at just a symbolic price).

Mike Kamrad
Runtime system for an application switch

- More efficient than tasking on pthreads (e.g. RTEMS)

An Application (Layer 7) Routing Switch with Ada95 Software

Mike Kamrad

Top Layer Networks, Inc.
2400 Computer Drive
Westborough MA 01581
USA
+1.508.870.1300 x139
kamrad@TopLayer.com

Abstract. The Top Layer Networks AppSwitch™ is a coordinated hardware and software Layer 7-application switch designed to provide Application Control for data communication networks by automatically prioritizing network traffic according to the user and the application that is generating the messages. Ada was chosen as the programming language for AppSwitch™ software because it possessed the best combination of language features to provide both high reliability and portability, specifically, language. The AppSwitch™ is a multiprocessor system and the software architecture is designed to take advantage of Ada's distributed system features as Distributed System Annex implementations mature. Top Layer faced significant obstacles to make Ada succeed: a huge learning curve, an incomplete Ada tool chain for the processors in the system and the construction of the Ada runtime system to efficiently support the Ravenscar.

1 Introduction

Top Layer Networks (formerly Top Layer) is a new data communication product company, whose new product, the AppSwitch™ 2000, uses Ada as its implementation language. Ada is not the traditional programming language for building data communication product software. As such, Top Layer had to overcome significant obstacles to realize its full benefits. The goal of this paper is to describe the software architecture of the AppSwitch™ and how well Ada supported the architecture and to reflect on the Top Layer experience in effectively using Ada and overcoming obstacles to its effective use. The paper is divided as follows: Section 2 gives a brief overview of the AppSwitch™ 2000, in particular the software architecture of the AppSwitch™ 2000. Section 3 describes why Ada was chosen, an overview of the software development environment and how Ada features enhanced that software architecture. Section 4 describes the obstacles Top Layer faced and the lessons learned.

M. González Harbour and J.A. de la Puente (Eds.): Ada-Europe'99, LNCS 1622, 250-262, 1999.

The Ravenscar profile

- 8th IRTAW 1997
  Ravenscar, Yorkshire, UK

  A. Burns, A. Wellings
  Restricted tasking models
  Ada Letters, XVIII, 5, 1997

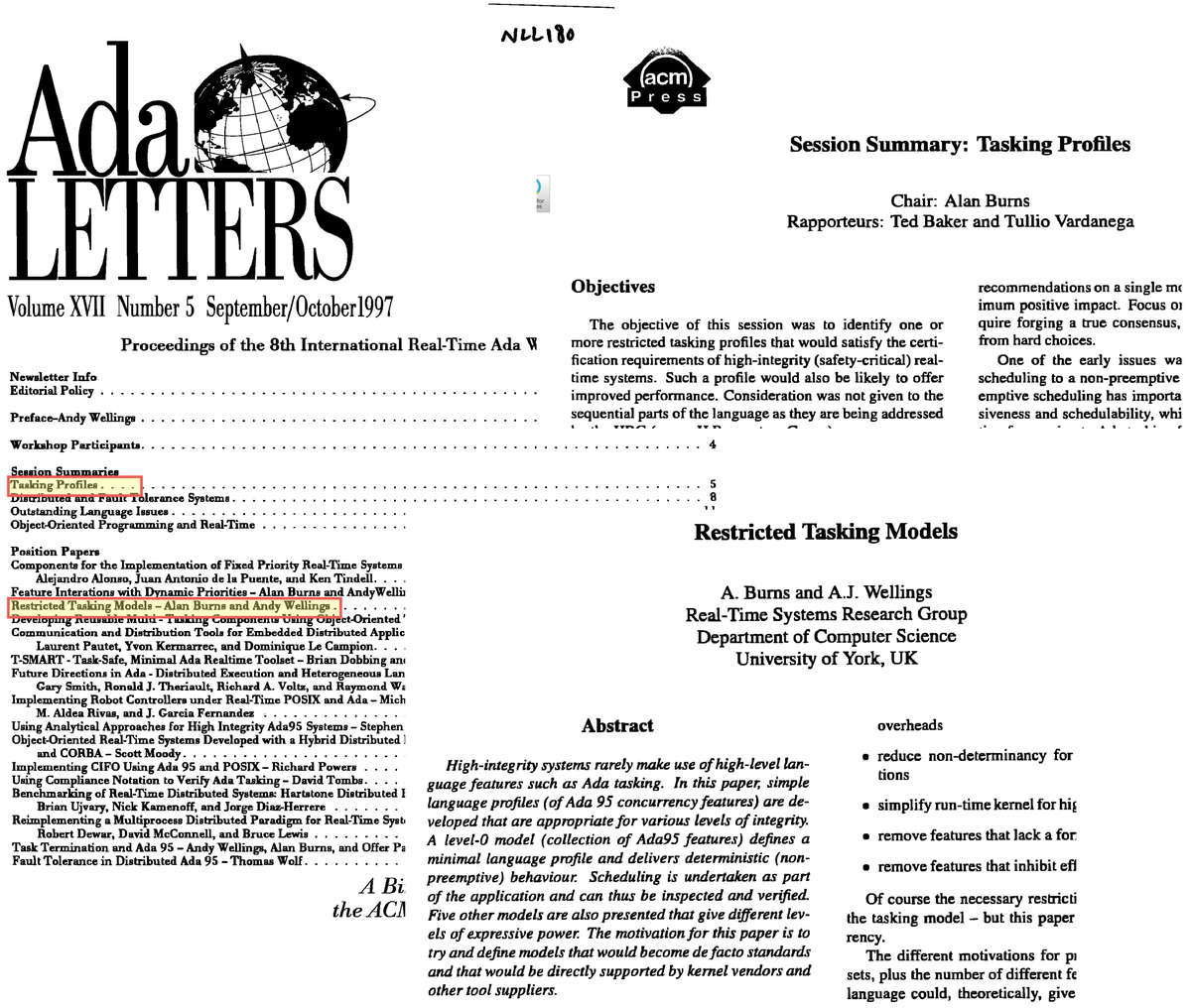
  Session summary: **Tasking profiles**
  A. Burns, T. Baker, T. Vardanega
  Ada Letters, XVIII, 5, 1997

- Revised and refined at SIGAda'98

  B. Dobbing, A. Burns
  The Ravenscar tasking profile for high integrity real-time programs
  Ada Letters, XVIII, 6. 1998

  and Ada-Europe 98

  A. Burns, B. Dobbing & G. Romanski
  The Ravenscar tasking profile for high integrity real-time programs
  LNCS 1411



Ada LETTERS

Volume XVII Number 5 September/October1997

Proceedings of the 8th International Real-Time Ada W

Newsletter Info
Editorial Policy . . . .

Preface-Andy Wellings . . . .

Workshop Participants 4

Session Summaries
Tasking Profiles 5
Distributed and Fault Tolerance Systems 8
Outstanding Language Issues . . . .
Object-Oriented Programming and Real-Time . . . .

Position Papers
Components for the Implementation of Fixed Priority Real-Time Systems
Alejandro Alonso, Juan Antonio de la Puente, and Ken Tindell. . . .
Feature Interations with Dynamic Priorities – Alan Burns and AndyWelli
Restricted Tasking Models – Alan Burns and Andy Wellings . . . .
Developing Reusable Multi - Tasking Components Using Object-Oriented
Communication and Distribution Tools for Embedded Distributed Applic
Laurent Pautet, Yvon Kermarrec, and Dominique Le Campion. . . .
T-SMART - Task-Safe, Minimal Ada Realtime Toolset – Brian Dobbing an
Future Directions in Ada - Distributed Execution and Heterogeneous Lan
Gary Smith, Ronald J. Theriault, Richard A. Voltz, and Raymond W
Implementing Robot Controllers under Real-Time POSIX and Ada – Mich
M. Aldea Rivas, and J. Garcia Fernandez . . . .
Using Analytical Approaches for High Integrity Ada95 Systems – Stephen
Object-Oriented Real-Time Systems Developed with a Hybrid Distributed
and CORBA – Scott Moody . . . .
Implementing CIFO Using Ada 95 and POSIX – Richard Powers . . . .
Using Compliance Notation to Verify Ada Tasking – David Tombs . . . .
Benchmarking of Real-Time Distributed Systems: Hartstone Distributed
Brian Ujvary, Nick Kamenoff, and Jorge Diaz-Herrere . . . .
Reimplementing a Multiprocess Distributed Paradigm for Real-Time Syst
Robert Dewar, David McConnell, and Bruce Lewis . . . .
Task Termination and Ada 95 – Andy Wellings, Alan Burns, and Offer Pa
Fault Tolerance in Distributed Ada 95 – Thomas Wolf . . . .

A Bi
the ACM

NLL180

(acm) Press

Session Summary: Tasking Profiles

Chair: Alan Burns
Rapporteurs: Ted Baker and Tullio Vardanega

Objectives

The objective of this session was to identify one or more restricted tasking profiles that would satisfy the certification requirements of high-integrity (safety-critical) real-time systems. Such a profile would also be likely to offer improved performance. Consideration was not given to the sequential parts of the language as they are being addressed

recommendations on a single mo
imum positive impact. Focus o
quire forging a true consensus,
from hard choices.

One of the early issues wa
scheduling to a non-preemptive
emptive scheduling has importa
siveness and schedulability, whi

Restricted Tasking Models

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

Abstract

High-integrity systems rarely make use of high-level language features such as Ada tasking. In this paper, simple language profiles (of Ada 95 concurrency features) are developed that are appropriate for various levels of integrity. A level-0 model (collection of Ada95 features) defines a minimal language profile and delivers deterministic (non-preemptive) behaviour. Scheduling is undertaken as part of the application and can thus be inspected and verified. Five other models are also presented that give different levels of expressive power. The motivation for this paper is to try and define models that would become de facto standards and that would be directly supported by kernel vendors and other tool suppliers.

overheads

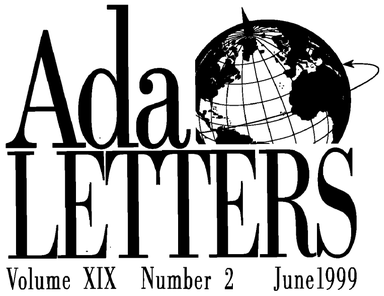
- reduce non-determinancy for
  tions
- simplify run-time kernel for hig
- remove features that lack a for
- remove features that inhibit efl

Of course the necessary restricti
the tasking model – but this paper
rency.

The different motivations for pr
sets, plus the number of different fe
language could, theoretically, give

A first attempt: JTK

- José's Tasking Kernel (José Ruiz' Master)
  - a tasking kernel for GNAT running on ix86
  - not yet fully Ravenscar
- 9th IRTAW 1999
  Wakulla Springs Lodge, FL, USA
  J.A. de la Puente, J.F. Ruiz, J.M. Gonzalez-Barahona
  Real-time programming with GNAT: specialised kernels versus POSIX threads
  Ada Letters XIX - 2, pp. 73 - 77. 1999.
- ESA (Tullio Vardanega) showed interest in
  - porting to ERC32 (SPARC) architecture
  - Ravenscar compliance



Volume XIX Number 2 June1999

Table of Cont

Meetings
SIGAda '99 .....
10th International Real-Time Ada Workshop – Call for Participation .....

Proceedings of the 9th International Real-Time Ada Workshop .....
Preface – Alan Burns .....
Workshop Participants .....

Session Summaries
Fault Tolerance .....
The Ravenscar Profile and Implementation Issues .....
Distributed Ada and Real-Time .....
New Language Features and other Language Issues .....

Position Papers
Replica Management in Real-Time Ada 95 Applications – L. M. Pinho a
The Ravenscar Tasking Profile – Experience Report – B. Dobbing and G
Transparent Replication for Fault Tolerance in Distributed Ada 95 – T. Wolf 33
Dynamic Ceiling Priorities and Ada 95 – J. Real and A. Wellings 41
Combining Tasking and Transactions – J. Kienzle 49
Extendable, Dispatchable Task Communication Mechanisms – S. Michell and K. Lundqvist 54
SimpleGraphics: Tcl/Tk Visualization of Real-Time Multi-Threaded and Distributed Application – S. A. Moody, S. Kwok, and D. Karr 60
Prioritizing Remote Procedure Calls in Ada Distributed systems – J. J. G. Garcia, M. G. Harbour 67
Real-Time Programming with GNAT: Specialised Kernels versus POSIX Threads – J. A. de la Puente, J. F. Ruiz, and J. M. Gonzalez-Barahona 73
How to Verify Concurrent Ada Programs: The Application of Model Checking – A. Burns and A. J. Wellings 78
An Experimental Testbed for Embedded Real Time Ada 95 – W. M. Walker, P. T. Woolley, and A. Burns 84
Distributed Programming with Intermediate IDL – G. W. Smith and R. A. Volz 90
A Linux Kernel Module Implementation of Restricted Ada Tasking – H. Shen and T. P. Baker 96

Real-Time Programming with GNAT: Specialised Kernels versus POSIX Threads

Juan A. de la Puente[1], José F. Ruiz[1], and Jesús M. González-Barahona[2],
[1]Universidad Politécnica de Madrid
[2]Universidad Carlos III de Madrid
E-mail: jpuente@dit.upm.es, jfruiz@dit.upm.es, jgb@computer.org

Abstract

The fact that most of the GNAT ports are based on non real-time operating systems leads to a reduced usability for developing real-time systems. Otherwise, existing ports over real-time operating systems are excessively complex, since GNAT uses only a reduced set of their functionality, and with a very specific semantic. This paper describes the implementation of a low-level tasking support for the GNAT run-time. In order to achieve a predictable real-time behaviour we have developed a very simple library, built to fit only the GNAT tasking requirements. We have also designed a bare machine kernel which provides the minimum environment needed by the upper layers.

Keywords: *Ada-95, GNAT, run-time system, real-time kernels*

1. Introduction

The development of GNAT was a decisive step towards the widespread availability of an efficient, high quality compiling environment to Ada programmers. The fact that GNAT is free software is of great interest for researchers, since it allows new developments from existing source code.

Although GNAT provides an effective, high quality compiling environment for Ada 95, its usability for real-time systems development is limited, as most of the GNAT ports are based on non real-time operating systems. Although all GNAT ports implement most of the Annex C and D functionality, many important features, such as true pre-emptive priority scheduling, monotonic time, ceiling locking, and kernel metrics, are not provided as specified in the LRM. As a result, most GNAT implementations cannot be used to program real-time systems with a predictable behaviour.

Looking at GNAT ports over real-time operating systems, we can cite RTEMS[8], a free real-time executive with a POSIX interface and support for multiprocessor systems. But it has been designed for a generic use, and there is a big overhead and an excessive complexity when using it as low-level support for the GNAT tasking system.

The most common way of implementing GNARL[1] is on top of native threads (usually POSIX threads or Pthreads for short) for the given architecture. But GNAT tasking implementation is very complete and specific, and when implementing GNARL on top of Pthreads there is a high overhead motivated by the similar level of abstraction of Ada tasks and Pthreads[4]. Aside from the loss of performance, it increases the complexity, leading to a difficult measuring and bounding of the kernel metrics. Indeed, in the case of many embedded systems a full-blown implementation of Pthreads is usually considered to be too expensive, and then the existence of a reduced and simple thread support could be of great help.

Therefore, our purpose is to develop a very simple and efficient real-time support for the GNAT tasking system, adapted to its requirements. By not requiring support for the more complex thread features, this approach permits an implementation with very tight efficiency and timing predictability requirements. The library that implements the low level tasking (we call our library JTK from Jose's Tasking Kernel) provides GNARL semantics and is written in Ada[2]. The kernel that interacts with the underlying hardware is written in C, with a small amount of assembly code.

Our intention is to provide a freely available test-bed for experimentation in language, compiler, and run-time support for developers of real-time embedded systems.

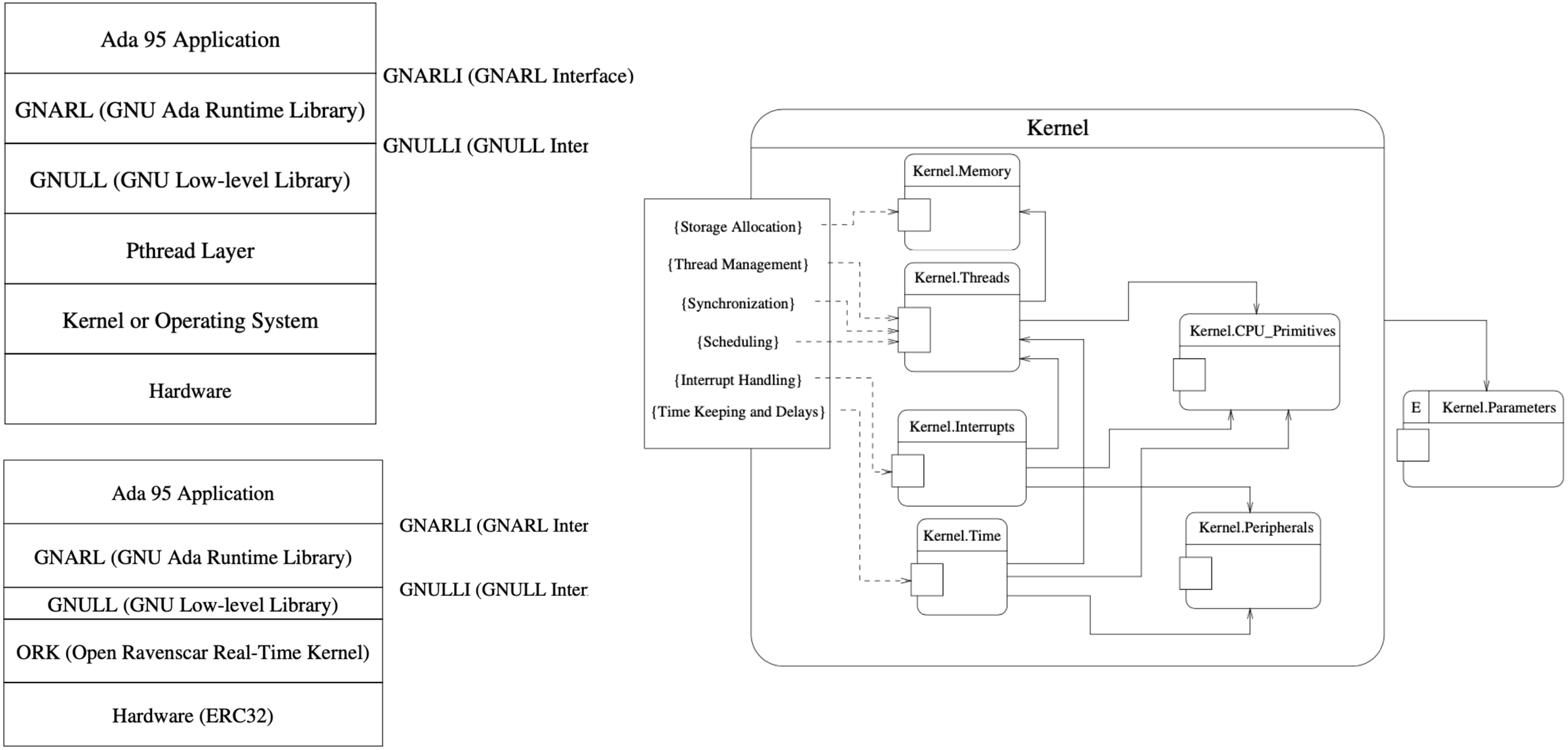
1. GNU Ada Runtime Library.
2. About 1000 lines of code, not including test programs.

The ORK project

- ESA contract 1999–2000
  - UPM (+ URJC)
  - CASA
  - U. York
- GNAT/ORK 1.0 released in June 2000
  - hosted at GNU/Linux, targeted at ERC32 (SPARC v7)
  - developed according to ESA software standards (ECSS-E-40)
- Further extensions
  - porting to other platforms (LEON & XtratuM)
  - verification & metrics
  - application examples (OBOSS)
- Eventually merged with ACT's GNAT for LEON

... and went into space

ORK architecture



First dissemination papers

ORK: An Open Source Real-Time Kernel for On-Board Software Systems*

Juan A. de la Puente José F. Ruiz Juan Zamorano Rodrigo García
Ramón Fernández-Marina

Departamento de Ingeniería de Sistemas Telemáticos
ETSIT, Universidad Politécnica de Madrid
Ciudad Universitaria, E-28040 Madrid, Spain
Phone +34 9 13 36 73 42; Fax +34 9 13 36 73 33

Abstract

Ada tasking is a powerful abstraction mechanism for developing concurrent systems. However, many implementations of concurrent tasking have been seen as potentially unsafe for critical systems because of their high degree of indeterminism. The *Ravenscar profile* is a subset of Ada 95 tasking with purpose of providing a basis for the implementation of certifiable critical systems. ORK is an open-source real-time kernel which provides full conformance with the Ravenscar profile on ERC32 computers. The kernel has a reduced size and complexity, and has been carefully designed to allow the building of reliable software for on-board space applications. This kernel is integrated in a cross-compilation system based on GNAT 3.13, including a tasking-aware version of GDB.

1 Introduction

Mission-critical on-board software has usually been developed on top of a cyclic executive that invokes the execution of application tasks according to a predefined static schedule. There are thus no concurrent threads of execution, and the application code is made of a set of purely sequential procedures.

This approach leads to simple, robust implementations, and provides a deterministic time behaviour which has often been considered a requirement for critical real-time systems. However, as the functionality and complexity of on-board software increases, and there is more and more pressure for shortening development times and reducing costs, while keeping up with critical reliability requirements, its low-level nature and lack of flexibility make it less appropriate. As a consequence, more attention is being devoted to higher level, abstract development methods that include concurrency as a means of decoupling application tasks and making software easier to design and test [20].

Indeed, many implementations of concurrent tasking have been seen as potentially unsafe for critical systems because of their high degree of indeterminism, which may make programs difficult to validate. This has led to either completely banning tasking for critical software applications, which is the traditional approach, or to the more flexible approach of building specialised kernels with reduced functionality. By limiting the way tasks are executed and synchronized, it can be expected that concurrent systems can be analysed and tested, so that safe concurrent systems can be built in a way that has significant advantages over cyclic executives from the point of view of flexibility and structuring [13].

Ada [1] is the language of choice for many critical systems due to its careful design and the existence of clear guidelines for building safe systems [12]. While the first approaches to developing safe Ada software did not make use of Ada tasking [10, 4], recent advances in real-time systems timing analysis methods [2] have paved the way to safe tasking in Ada. The *Ravenscar profile* [3, 6] is a subset of Ada 95 tasking that was defined at the 8th International Real-Time Ada Workshop (IRTAW8) with purpose of providing a basis for the implementation of certifiable critical systems. The first implementation of the profile, Aonix' *Raven*, has indeed shown the feasibility of the approach and the possibility of building certifiable applications based on it [8].

Based on this early experience, the European Space Research and Technology Centre (ESTEC) launched the *Open Ravenscar Real-time kernel (ORK)* project in September 1999. The aim of the project is to develop an open-source kernel, compliant with the Ravenscar profile, for its current standard on-board computer, ERC-32, which is a radiation-hardened

*The work described in this paper is being carried out under ESA/ESTEC contract no. No.13863/99/NL/MV.

DASIA 2000 (Montreal)

An Open Ravenscar Real-Time Kernel for GNAT*

Juan A. de la Puente[1], José F. Ruiz[1], and Juan Zamorano[2]

[1] Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid, E-28040 Madrid, Spain
jpuente@dit.upm.es, jfruiz@dit.upm.es
[2] Departamento de Arquitectura y Tecnología de Sistemas Informáticos
Universidad Politécnica de Madrid, E-28660 Madrid, Spain
jzamora@datsi.fi.upm.es

Abstract. This paper describes the architecture of ORK, an open source real-time kernel that implements the Ravenscar profile for the GNAT compilation system on a bare ERC32 computer. The kernel has a reduced size and complexity, and has been carefully designed in order to make it possible to build reliable software for on-board space applications. The kernel is closely integrated with the GNAT runtime library, and supports Ada tasking in an efficient and compact way.

1 Introduction

The Ravenscar Profile [4,6] is the best known result of the 8th International Real-Time Ada Workshop (IRTAW8). It defines a subset of the tasking features of Ada which can be implemented using a small, reliable kernel. The expected benefits of this approach are:

- Improved memory and execution time efficiency, by removing features with a high overhead.
- Improved reliability, by removing non-deterministic and non analysable features.
- Improved timing analysis, by removing non-deterministic and non-analysable features.

The tasking model defined by the profile includes tasks and protected types and objects at the library level, a maximum of one protected entry with a simple boolean barrier for synchronization, a real-time clock, absolute delays, preemptive priority scheduling with ceiling locking access to protected objects, and protected procedure interrupt handlers, as well as some other features. Other features, such as dynamic tasks and protected objects, task entries, dynamic priorities, select statements, asynchronous transfer of control, relative delays, or

* This work has been funded by ESA/ESTEC contract no. No.13863/99/NL/MV.

H. B. Keller and E. Plödereder (Eds.): Ada-Europe 2000, LNCS 1845, pp. 5–15, 2000.

Ada-Europe 2000 (Potsdam)

The Design and Implementation of the Open Ravenscar Kernel*

Juan A. de la Puente Juan Zamorano José Ruiz Ramón Fernández Rodrigo García

Department of Telematics Engineering
Technical University of Madrid, Spain
E-mail: jpuente@dit.upm.es

Abstract

This paper describes the design and implementation of the Open Ravenscar Kernel (ORK), an open-source real-time kernel of reduced size and complexity, for which users can seek certification for mission-critical space applications. The kernel supports Ada 95 tasking on an ERC32 (SPARC v7) architecture in an efficient and compact way. It is closely integrated with the GNAT runtime library and other tools.

1. Introduction

The Open Ravenscar Real-Time Kernel (ORK) [10, 11] is a tasking kernel for the Ada language [2] which provides full conformance with the Ravenscar profile [6, 4, 7] on ERC32-based computers. ERC32 is a radiation-hardened implementation of the SPARC V7 architecture, which has been adopted by the European Space Agency (ESA) as the current standard processor for spacecraft on-board computer systems [12].

ORK supports the restricted version of Ada tasking defined by the Ravenscar profile, which includes static tasks (with no entries) and protected objects (with at most one entry), a real-time clock and delay until statements, and protected interrupt handler procedures, as well as other tasking features.

The kernel is fully integrated with the GNAT compilation system. Debugging support for the ORK kernel, including tasking, is based on an enhanced version of the GDB debugger and the DDD graphic front-end. The distribution includes an adapted version of GNAT hosted on GNU/Linux workstations and targeted to ERC32 bare computers, the kernel itself, adapted version of GDB and DDD, and some additional libraries and tools. It is freely available as an open source product, with a GPL license[1].

*This work has been funded by ESA/ESTEC contract no. No.13863/99/NL/MV.

[1]ORK and its associated software can be downloaded from http://www.openravenscar.com

This paper describes the design and implementation of ORK. The rest of the paper is organised as follows: Section 2 describes how the Ravenscar profile can be implemented in GNAT. Section 3 describes the ORK design and section 4 deals with some implementation issues. Finally, some conclusions and plans for the near future are included in section 5.

2. Support for the Ravenscar profile in GNAT

2.1. Compile-time checking

Most of the Ada subset defined by the Ravenscar profile can be checked at compile time by using an appropriate set of restriction identifiers with the pragma Restrictions (ALRM, D.7, H.4). However, not all the Ravenscar restrictions can be enforced by standard identifiers, and thus a number of additional restriction identifiers have been proposed at the last IRTAW meetings in order to support the profile [7].

The most recent versions of GNAT (from 3.12 on) have included most of the non-standard Ravenscar restrictions as implementation-specific pragmas. However, there are a couple of restrictions that are not implemented in GNAT or are implemented in a slightly different way than specified by the profile:

- The Ravenscar restriction Simple_Barrier_Variables is replaced in GNAT by Boolean_Entry_Barriers. The semantics of this restriction is the same as the original one.
- The Ravenscar restriction Max_Entry_Queue_Depth => N (with $N = 1$ for Ravenscar compliant programs) is replaced in GNAT by No_Entry_Queue. In this case, the semantics is the same, but the restriction name is somewhat misleading, as there may still be one task waiting on an entry barrier to be opened (i.e. a queue with just one task).

IRTAW10 (´Las Navas del Marqués)

Project team

UPM

Juan A. de la Puente

Juan Zamorano

Alejandro Alonso

José Francisco Ruiz

Ramón Fernández Marina

Rodrigo García

Miguel Ángel Ajo

Ángel Álvarez

URJC

Jesús González Barahona

Vicente Matellán

Andrés Arias

Juan Manuel Dodero

José Centeno

Pedro de las Heras

CASA

Andrés Borges

Juan Carlos Morcuende

Jesús Borruel

University of York

Alan burns

Andy Wellings

ESA/ESTEC

Jean-Loup Terraillon

Jorge Amador

Tullio Vardanega

POLITÉCNICA