CONCURRENT AND REAL-TIME PROGRAMMING IN ADA

Ada is the only ISO standard, object-oriented, concurrent, real-time programming language. It is intended for use in large, long-lived applications where reliability and efficiency are essential, particularly real-time and embedded systems. In this book, Alan Burns and Andy Wellings give a thorough, self-contained account of how the Ada tasking model can be used to construct a wide range of concurrent and real-time systems. This is the only book that focuses on an in-depth discussion of the Ada tasking model. Following on from the authors' earlier title 'Concurrency in Ada', this book brings the discussion up to date to include the new Ada 2005 language and the recent advances in real-time programming techniques. It will be of value to software professionals and advanced students of programming alike; indeed, every Ada programmer will find it essential reading and a primary reference work that will sit alongside the language reference manual.

ALAN BURNS is a Professor in Computer Science at the University of York. His research activities have covered a number of aspects of real-time and safety critical systems, including the assessment of languages for use in the real-time safety critical domain, distributed operating systems, the formal specification of scheduling algorithms and implementation strategies, and the design of dependable user interfaces to safety critical applications. His teaching activities include courses in Operating Systems, Scheduling and Real-time Systems. He has authored over 370 papers and reports and 8 books, including 'Real-time Systems and Programming Languages' (3rd Edition), 'Concurrency in Ada' (2nd Edition) and 'Concurrent and Real-Time Programming in Java'.

ANDY WELLINGS is a Professor of Real-Time Systems in the Computer Science Department at the University of York. He is interested in most aspects of the design and implementation of real-time dependable computer systems and, in particular, in real-time programming languages and operating systems. He is European Editor-in-Chief for the Computer Science journal 'Software-Practice and Experience' and a member of the International Expert Groups currently developing extensions to the Java platform for real-time, safety critical and distributed programming. He has authored over 280 papers and several books, including 'Real-time Systems and Programming Languages' (3rd edition) and 'Concurrency in Ada' (2nd edition).

CONCURRENT AND REAL-TIME PROGRAMMING IN ADA 2005

ALAN BURNS AND ANDY WELLINGS
University of York

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521866972

First published 2007

Printed in the United Kingdom at the University Press, Cambridge

A catalogue record for this publication is available from the British Library

ISBN 978-0-521-86697-2 hardback

Contents

Preface

The development of the Ada programming language forms a unique and, at times, intriguing contribution to the history of computer languages. As all users of Ada must know, the original language design was a result of competition between a number of organisations, each of which attempted to give a complete language definition in response to a series of documented requirements. This gave rise to Ada 83. Following 10 years of use, Ada was subject to a complete overhaul. The resulting language, Ada 95, had a number of significant changes from its predecessor. A further 10 years of use has produced another version of Ada, known as Ada 2005, this time the changes are less pronounced and yet there are some key extra facilities, especially in the areas of real-time programming.

Closely linked to the development of Ada has been this book on its concurrent features. Starting out as 'Concurrent Programming in Ada', it became 'Concurrency in Ada' when the Ada 95 version of the language was defined. There were two editions of this title. With the new features of Ada 2005, it has been decided to broaden the focus of the book to include real-time issues – hence this first edition of the new title 'Concurrent and Real-Time Programming in Ada 2005'. No prior knowledge of concurrent programming (in general) or of Ada tasking (in particular) is assumed in this book. However, readers should have a good understanding of at least one high-level sequential programming language and some knowledge of operating system principles.

This book is aimed both at professional software engineers and at students of computer science (and other related disciplines). Many millions of lines of Ada 83 and 95 code have been produced world wide, and over the next decade a wide range of new applications will be designed with Ada 2005 as the target language. It is important that Ada programmers do not restrict themselves to a sequential subset of the language on the dubious assumption that tasking is not appropriate to their work, or for fear that the tasking model is too complex and expensive. Tasking is an integral part of the language, and programmers must be familiar with,

if not experienced in, its use. Due to space considerations, books that describe the entire language may not deal adequately with the tasking model; this book therefore concentrates exclusively on this model.

Students studying real-time programming, software engineering, concurrent programming or language design should find this book useful in that it gives a comprehensive description of the features that one language provides. Ada is not merely a product of academic research (as are many concurrent programming languages) but is a language intended for actual use in industry. Its model of tasking was therefore integrated into the entire language design, and the interactions between tasking and non-tasking features were carefully defined. Consequently, the study of Ada's model of concurrency should be included in those advanced courses mentioned above. However, this does not imply that the full tasking model is free from controversy, has a proven formal semantic basis or is amenable to efficient implementation. The nature of these areas of 'discussion' are dealt with, as they arise in this book.

Unlike Ada 83, which defined a single language, the Ada 95 and 2005 definitions have a core language design plus a number of domain-specific annexes. A compiler need not support all the annexes but it must support the core language. Most of the tasking features are contained in the core definition. But there are relevant annexes that address systems programming and real-time programming.

The first chapter provides a basic introduction to concurrent and real-time systems and gives an overview of the clock facilities within Ada.

Chapters 2 and 3 look in detail at the uses of concurrent programming and the inherent difficulties of providing inter-process communication. There is, as yet, no agreement on which primitives a concurrent programming language should support and, as a consequence, many different styles and forms exist. In order to understand the Ada tasking model fully, it is necessary to appreciate these different approaches and the problems faced by the user of any language that supports multi-processing.

The Ada task is introduced in Chapter 4 and the rendezvous and the important select statement are considered in the following two chapters. The rendezvous provides a synchronous communication mechanism. Data-orientated asynchronous communication is considered in Chapter 7, together with the important abstraction of a protected object. This provides a passive means of encapsulating data and providing mutual exclusion. An effective way of increasing the expressive power of the communication primitives is the requeue facility. This is described, with many examples given, in Chapter 8. The relationship between tasks and exceptions is dealt with in Chapter 9. This chapter also covers the means by which one task can affect the behaviour of another task asynchronously.

Chapter 10 considers the interplay between tasking and the object-orientated programming features of the language. This forms the basis from which a collec-

tion of concurrency utilities can be defined. A number of these are provided in Chapter 11.

As indicated earlier, a number of the annexes deal with issues of relevance to concurrent programming. Chapter 12 considers systems programming (including support for low level programming). For real-time programmers, perhaps the most important issue is scheduling. Ada provides a comprehensive list of features that are covered in Chapters 13 and 14. In addition to scheduling, real-time programs need to have control over when events are executed and control over the resources that tasks and groups of task require at run-time. These issues are covered in Chapter 15.

Having introduced all of Ada's relevant features, Chapter 16 then provides a collection of real-time utilities that can be used to gain access to the power of the language. This is followed in Chapter 17 by consideration of the usefulness of subsetting Ada and using profiles to gain access to efficient and certifiable implementations. In particular, the Ravenscar profile is described in this chapter. Finally, in Chapter 18 conclusions are provided and a short summary of the differences between Ada 2005 and Ada 95 is given in the context of concurrent and real-time programming, together with a brief look to the future.

The material presented in this book reflects the authors' experiences in both using and teaching Ada tasking. Teaching experience has been obtained by writing and presenting courses at the University of York (UK) and by developing educational material and training.

Further material

Further material on all aspects of real-time and concurrency in Ada 2005 can be found on a www page dedicated to this book:
http://www.cs.york.ac.uk/∼rts/ada/CRTIA.html.

Real-time systems research at York

Alan Burns and Andy Wellings are members of the Real-Time Systems Research Group in the Department of Computer Science at the University of York (UK).

The aim of the group is to undertake fundamental research, and to bring modern techniques, methods and tools into engineering practice. Areas of application of our work include space and avionic systems, engine controllers, automobile control, manufacturing systems, sensor nets and multi-media systems. Specifically, the group is addressing: scheduling theories, language design, kernel design, communication protocols, distributed and parallel architectures and program code analysis.

Further information about the group's activities can be found via our www page: http://www.cs.york.ac.uk/∼rts

Acknowledgements

The authors would like to thank the following individuals who have, directly or indirectly, helped during the preparation of this book. In particular, John Barnes, Andrew Borg, Javier Miranda, Pat Rogers, Ed Schonberg, and Tucker Taft.

The authors also wish to thank members of ISO committee ARG and attendees of the IRTAW series for their input to some of the issues discussed in this book.

Finally, we would like to thank the AdaCore GNAT project team for all their efforts to produce a public domain Ada 2005 compiler. Writing a book for a new language is very difficult when there are no validated compilers to help test the code. Access to the GNAT system has provided us with more confidence that the code given in this book is at least syntactically correct!