ADA USER JOURNAL

Volume 29
Number 2
June 2008

Contents

Editorial Policy for Ada User Journal

Publication

Ada User Journal — The Journal for the international Ada Community — is published by Ada-Europe. It appears four times a year, on the last days of March, June, September and December. Copy date is the last day of the month of publication.

Aims

Ada User Journal aims to inform readers of developments in the Ada programming language and its use, general Ada-related software engineering issues and Ada-related activities in Europe and other parts of the world. The language of the journal is English.

Although the title of the Journal refers to the Ada language, any related topics are welcome. In particular papers in any of the areas related to reliable software technologies.

The Journal publishes the following types of material:

- Refereed original articles on technical matters concerning Ada and related topics.
- News and miscellany of interest to the Ada community.
- Reprints of articles published elsewhere that deserve a wider audience.
- Commentaries on matters relating to Ada and software engineering.
- Announcements and reports of conferences and workshops.
- Reviews of publications in the field of software engineering.
- Announcements regarding standards concerning Ada.

Further details on our approach to these are given below.

Original Papers

Manuscripts should be submitted in accordance with the submission guidelines (below).

All original technical contributions are submitted to refereeing by at least two people. Names of referees will be kept confidential, but their comments will be relayed to the authors at the discretion of the Editor.

The first named author will receive a complimentary copy of the issue of the Journal in which their paper appears.

By submitting a manuscript, authors grant Ada-Europe an unlimited license to publish (and, if appropriate, republish) it, if and when the article is accepted for publication. We do not require that authors assign copyright to the Journal.

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

News and Product Announcements

Ada User Journal is one of the ways in which people find out what is going on in the Ada community. Since not all of our readers have access to resources such as the World Wide Web and Usenet, or have enough time to search through the information that can be found in those resources, we reprint or report on items that may be of interest to them.

Reprinted Articles

While original material is our first priority, we are willing to reprint (with the permission of the copyright holder) material previously submitted elsewhere if it is appropriate to give it a wider audience. This includes papers published in North America that are not easily available in Europe.
We have a reciprocal approach in granting permission for other publications to reprint papers originally published in *Ada User Journal.*

Commentaries

We publish commentaries on Ada and software engineering topics. These may represent the views either of individuals or of organisations. Such articles can be of any length – inclusion is at the discretion of the Editor.

Opinions expressed within the *Ada User Journal* do not necessarily represent the views of the Editor, Ada-Europe or its directors.

Announcements and Reports

We are happy to publicise and report on events that may be of interest to our readers.

Reviews

Inclusion of any review in the Journal is at the discretion of the Editor.
A reviewer will be selected by the Editor to review any book or other publication sent to us. We are also prepared to print reviews submitted from elsewhere at the discretion of the Editor.

Submission Guidelines

All material for publication should be sent to the Editor, preferably in electronic format. The Editor will only accept typed manuscripts by prior arrangement.
Prospective authors are encouraged to contact the Editor by email to determine the best format for submission. Contact details can be found near the front of each edition.
Example papers conforming to formatting requirements as well as some word processor templates are available from the editor. There is no limitation on the length of papers, though a paper longer than 10,000 words would be regarded as exceptional.

Editorial

This June issue of the Journal is being finalised shortly after the Ada-Europe conference, which took place in the beautiful Venice, the 16th to 20th of June. I would like to congratulate the organizers, for the rich and enjoyable conference, not only in the contents of the program, but also in the logistics of the conference. As announced during the conference, next year the conference will take place in Brest, France, in the second week of June. You can find the preliminary announcement in the Forthcoming Events section of this issue.

Another announcement that was put forward in Venice, and that I am pleased to second here, is the launch of the online archive of the Ada User Journal. This archive is still a work-in-progress, but already the Ada community may browse, consult and download contents of some of the back issues of the Journal. I invite all of you to the archives, reachable from the Journal section of the Ada-Europe webpage at www.ada-europe.org.

As for the current issue (its contents will be available in the online archives one year from now!); in it we publish two more sessions of the Proceedings of the 13th International Real-Time Ada Workshop. The first session provides insight on the implementation of the new real-time services in two differente kernels: ORK (a work coming from the Technical University of Madrid, Spain) and MarteOS (from the University of Cantabria, Spain). In the second session, the participants started to look to future direction for Ada, with a focus on multiprocessing in SMP systems and stream based parallel systems, with two papers coming from the University of York, UK.

To finalise, in the Ada Gems section we provide two gems on Ada and XML, by Pascal Obry and Emmanuel Briot. The News, Calendar and Forthcoming Events sections complete the issue.

In memory of Peter Amey.

Peter Amey passed away last April, shortly after the close of the March issue of the Journal. We express our condolences to his family and friends.

Luís Miguel Pinho
Porto
June 2008
Email: lmp@isep.ipp.pt

News

Santiago Urueña

Technical University of Madrid (UPM). Email: Santiago.Uruena@upm.es

Contents

Ada-related Organizations

ARA — Ada Does Multicore Now

From: Ada Information Clearinghouse
Subject: Ada Does Multicore Now
Date: March 19, 2008
URL: http://www.adaic.com/whyada/multicore.html

Programming multicore systems is a hot topic these days. As Robert Dewar notes in his EE Times commentary, "There's nothing new about multicore mania". Ada has been supporting multiprocessor, multicore, and multithreaded architectures as long it has existed. As he notes, "decades of experience have been accumulated in using Ada to deal with the problem of writing programs that run effectively on machines using more than one processor". For some reason, many pundits are ignoring all of this experience and clamoring for new ways to do multiprocessor programs. Indeed, Microsoft thinks that a programming model for multicore systems "will not emerge for five to 10 years", according to this Embedded.com article. [http://www.embedded.com/news/embeddedindustry/201200461]

While we're all waiting for that to happen, Ada practitioners will be using Ada to build high-performance multicore applications. For instance, Karl Nyberg of Grebyn Corporation won a Sun Fire T1000 server in Sun Microsystems Open Performance Contest by building a parallel application using Ada. No need to wait 10 years.

Overview of Ada's concurrency features

A concurrent program typically comprises active components that interact with each other either directly or through shared resources. Ada directly supports each of these elements. An active component is modeled by a task, Ada's unit of concurrent execution. Direct communication between two tasks is achieved by a rendezvous, which provides synchronous passing of data. A shared resource can be mapped to a protected object (which provides mutual exclusion) or to other less general features. These semantic building blocks are high-level enough to be appropriate for modeling the architecture of a concurrent program. Ada tasks (logical threads of execution) can be mapped to multiple processors, to multiple threads on a single processor (with or without hardware support), or to multiple cores of a single processor, depending on the compiler and target environment. The program need not change to run in any of these environments.

Resources on Ada's concurrency features

A brief overview of Ada tasks and related features (with examples) can be found in the Tasking section of the Ada Programming Wikibook. An equally brief introduction can be found in Chapter 19 of Ada 95: The Craft of Object Oriented Programming. Sections can be found in other free resources on Ada as well; see our free textbooks page for other examples. [http://www.adaic.com/free/freebook.html] The "bible" of Ada tasking is the book Concurrent and Real-Time Programming in Ada by Burns and Welling. It has recently been revised to include Ada 2005 features and recent research in the field.

[See also "There's nothing new about multicore mania — EE Times" in AUJ 29-1 (Mar 2008), p.17. —su]

ARA — Ada Helps Build Safe Systems

From: Ada Information Clearinghouse
Subject: Ada Helps Build Safe Systems
Date: May 16, 2008
URL: http://www.adaic.com/whatsnew.html

"Ada Helps Build Safe Systems" is a new page gathering together articles from the trade press on Ada and high-integrity systems.

[http://www.adaic.com/whyada/safety.html —su]

ARA — Enhance Security With Ada

From: Ada Information Clearinghouse
Subject: Enhance Security With Ada
Date: May 23, 2008
URL: http://www.adaic.com/whatsnew.html

"Enhance Security With Ada" is a new page gathering together articles from the trade press on Ada and security.

[http://www.adaic.com/whyada/security.html —su]

ARA — ACATS 3.0D

From: Ada Information Clearinghouse
Subject: UpDate: Ada Conformity Assessment Test Suite
Date: May 30, 2008
URL: http://www.adaic.com/whatsnew.html

ACATS Modification List 3.0D and the associated test files have been posted.

[See also "ARA — ACATS 3.0" in AUJ 29-1 (Mar 2008), p.5. —su]

Ada-related Events

[To give an idea about the many Ada-related events organized by local groups, some information is included here. If you are organizing such an event feel free to inform us as soon as possible. If you attended one please consider writing a small report for the Ada User Journal. —su]

May 18 — Ada-Belgium

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Subject: Ada-Belgium Spring 2008 Event, incl. Debian packaging workshop
Date: Sat, 3 May 2008 22:26:51 +0200 (CEST)
Organization: Ada-Belgium, c/o Dept. of Computer Science, K.U.Leuven
Newsgroups: comp.lang.ada,fr.comp.lang.ada

Ada-Belgium Spring 2008 Event
Sunday, May 18, 2008, 12:00-19:00
Leuven, Belgium
including at 14:00
2008 Ada-Belgium General Assembly
and at 15:00
Workshop on Creating Debian Packages of Ada Software
<http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/events/local.html>

Announcement

The next Ada-Belgium event will take place on Sunday, May 18, 2008 in Leuven.

The Ada-Belgium Board decided to propose a more interactive and social

format than our traditional evening events based on a General Assembly followed by a technical presentation. Therefore, this event starts at noon, runs until 7pm, and includes a barbecue, a key signing party, the 15th General Assembly of the organization, and a workshop on packaging Ada software for Debian hosted by Ludovic Brenta, principal maintainer of Ada in Debian.

Schedule

- 12:00 welcome and getting started (setting up computers and preparing food — please be there!)
- 13:00 barbecue
- 14:00 Ada-Belgium General Assembly
- 14:45 key signing party
- 15:00 workshop on creating Debian packages of Ada software
- 19:00 end

Participation

Everyone interested (members and non-members alike) is welcome at any or all parts of this event.

For practical reasons registration is required. If you would like to attend, please send an email before Tuesday, May 13, to Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be> with the subject "Ada-Belgium Spring 2008 Event", so you can get precise directions to the place of the meeting.

If you are a member but have not renewed your affiliation yet, please do so by paying the appropriate fee before the General Assembly (you have also received a printed request via normal mail). If you are interested to become a new member, please register by filling out the 2008 membership application form[1] and by paying the appropriate fee before the General Assembly. After payment you will receive a receipt from our treasurer and you are considered a member of the organization for the year 2008 with all member benefits[2]. Early renewal ensures you receive the full Ada-Belgium membership benefits (including the Ada-Europe indirect membership benefits package).

As mentioned at earlier occasions, we have a limited stock of documentation sets and Ada related CD-ROMs that were distributed at previous events. Most important are back issues of the Ada User Journal[3]. These will be available on a first-come first-serve basis at the General Assembly for current and new members.

[1] http://www.cs.kuleuven.be/~dirk/ada-belgium/forms/member-form08.html

[2] http://www.cs.kuleuven.be/~dirk/ada-belgium/member-benefit.html

[3] http://www.ada-europe.org/journal.html

Barbecue

The organization will provide food and beverage to all Ada-Belgium members. Non-members who want to participate at the barbecue are also welcome: they can choose to join the organization or pay the ludicrous sum of TEN EUROS per person to the Treasurer of the organization.

General Assembly

All Ada-Belgium members have a vote at the General Assembly, can add items to the agenda, and can be a candidate for a position on the Board[4]. See the separate official convocation[5] for all details.

[4] http://www.cs.kuleuven.be/~dirk/ada-belgium/board/

[5] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/08/080518-abga-conv.html

Key Signing Party

Wouldn't it be nice if a majority of people used GPG to sign their email every day so that you could send all non-signed email into the spam bin? To make that dream come true, please join and expand the global Web of Trust! [6]

What you should bring with you:

- an official ID card issued by your national government;
- your GPG key fingerprint (i.e. the output of gpg --fingerprint) on small paper slips; a dozen copies or so should be enough.

What you will go home with:

- signatures from all other participants;
- automatic inclusion in the global Web of Trust;
- the ability to digitally sign or encrypt anything you like.

[6] http://en.wikipedia.org/wiki/Web_of_Trust

Workshop: Packaging Ada Software for Debian

Debian[7], "The Universal Operating System", is simply the best platform for the enthusiast Ada developer. The features that distinguish Debian from the rest are:

- a binary distribution that avoids the need to recompile Florist, ASIS, GtkAda and all other Ada packages;
- a large number of packages intended for Ada developers;
- a clear and consistent policy[8] making all packages integrate seamlessly and interoperate;
- outstanding support for the Ada part of the GNU Compiler Collection (GCC) with unique innovations like libgnatvsn and libgnatprj not found anywhere else;
- backports of bug fixes from the bleeding edge of GCC development into the safe and stable compiler used for all Debian packages;
- support for more hardware architectures than any other Ada distribution: alpha, amd64, hppa, i386, ia64, kfreebsd-i386, powerpc, s3980 and sparc (with mips, mipsel and ppc64 added recently).
- a choice between "stable", "testing" and "unstable" versions of Debian to suit personal preferences;
- Debian is the mother of Ubuntu, Knoppix and dozens of other distributions which sometimes incorporate the Ada packages.

The goal of the workshop is to help people participate in this effort to bring even more Ada software to Debian, or to help maintain the existing packages.

What you should bring with you:

- your computer, already installed with Debian unstable or with an unstable chroot already created (see below);
- network cables (or WiFi already configured);
- monitor and keyboard, if your computer is not a laptop;
- power cables;
- some Ada software you would like to see in Debian but is not there (not necessarily software that you wrote; any software with a license permitting redistribution in source and binary form will do).

Note 1: if your computer does not run Debian as its main operating system, you can install Debian in a virtual machine (VMWare or other), in a jail on a FreeBSD system (Debian kfreebsd-i386), or in a chroot on any other distribution. Danny Beullens will offer help and assistance to those who would like to install Debian in a VMWare virtual machine.

Note 2: if you would like to install Debian as your main operating system but are uncomfortable doing so by yourself, please get in touch with your nearest Linux User Group (e.g. http://www.bxlug.be in Brussels).

What Ludovic Brenta will do for you:

- set up a local Debian mirror, so you can install or upgrade packages necessary for Ada package development;
- explain how to package Ada software for Debian;
- help you package your own program or library;
- answer questions about GNAT, GCC, Debian, etc.;
- if your package is suitable for inclusion in Debian, sponsor it for you.

What you will go home with:

- your own .deb packages installed on your computer;

- better understanding of how packaging works;
- better understanding of the Debian Policy for Ada;
- if your package is suitable, your name on the Debian Package Tracking System and your package on the next Debian DVD or CD-ROM distribution.

[7] http://www.debian.org/

[8] http://www.ada-france.org/debian/debian-ada-policy.html

Directions

To permit this more interactive and social format, the event takes place at private premises in Leuven. As instructed above, please inform us by e-mail if you would like to attend, and we'll provide you precise directions to the place of the meeting. Obviously, the number of participants we can accommodate is not unlimited, so don't delay...

Looking forward to meet many of you in Leuven!

[See also “Jun 12 — Ada-Belgium” in AUJ 28-2 (Jun 2007), pp.70–71. —su]

Jun 16–20 — Ada-Europe 2008

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Subject: Press Release — Reliable Software Technologies, Ada-Europe 2008
To: Ada-Europe-attendees@cs.kuleuven.be
Date: Sun, 8 Jun 2008 21:44:40 +0200 (MEST)

FINAL Call for Participation

13th International Conference on

Reliable Software Technologies — Ada-Europe 2008

16 – 20 June 2008, Venice, Italy

http://www.ada-europe.org/conference2008.html

Press release:

Ada-Europe Conference on Reliable Software Technologies

International experts meet in Venice

Venice (8 June 2008 21:00) — Ada-Europe, in cooperation with ACM's Special Interest Group on Ada, organizes the “13th International Conference on Reliable Software Technologies — Ada-Europe 2008” from 16 to 20 June in Venice, Italy.

The conference offers two days of tutorials, a full technical program of refereed papers, a collection of industrial presentations reflecting current practice and challenges, an educational track, three invited speakers, an industrial exhibition, and a social program.

The 10 excellent tutorials on Monday and Friday cover a broad range of topics: AADL — Architecture Analysis and Design Language, A Practical Introduction to Model-Driven Software Development using Eclipse, The Best of Ada 2005, Object-Oriented Programming in Ada 2005, Preserving Model-Asserted Properties at Run Time for High-Integrity Systems, Technical Basis of Model Driven Engineering, Languages for Safety-Critical Software — Issues and Assessment, Service-Oriented Architecture Concepts and Implementations, Verification Techniques for Dependable Systems, and Real-Time Scheduling Analysis of Ada Applications.

The technical program presents 20 refereed and carefully selected papers on the latest research, new tools, applications and industrial practice and experience, a collection of 11 industrial presentations reflecting current practice and challenges, and 4 presentations plus a panel discussion on Ada and Education. Springer Verlag publishes the proceedings of the conference, as LNCS Vol. 5026.

Three international experts present invited lectures on the topics: Embedded Software Design: Art or Science?, Lost in Translation, and Three Ways to Improve SOA Reliability.

The exhibition opens in the mid-morning break on Tuesday and runs continuously until the end of the afternoon break on Thursday. The exhibitors include the following vendors: AdaCore, Aonix, Ellidiss Software, Praxis High Integrity Systems, Rapita Systems, and Telelogic.

The social program includes on Tuesday evening a welcome reception at Palazzo Cavalli-Franchetti on the Grand Canal, accompanied by musical entertainment by members of the conference community, and on Wednesday evening a private boat trip along the Grand Canal to the conference banquet in the renowned “Osteria Ponte del Diavolo” restaurant at Torcello, the farthest island of the lagoon.

The conference takes place at the Centro Culturale Don Orione Artigianelli, at Zattere Dorsoduro 909/A, at the south end of Venice, some 15 minutes of leisurely walk from Piazza San Marco, perhaps the most renowned spot of the city. The full program is available on the conference web site. Registration is still open.

Latest updates:

- The “Final Program” is available on the conference web site <http://www.ada-europe.org/conference2008.html> and directly at <http://www.math.unipd.it/ae2008/final_program.pdf>.
- Check out the 10 tutorials in the final program and at <http://www.math.unipd.it/ae2008/protus.html>.
- The proceedings, published by Springer Verlag as Lecture Notes in Computer Science Vol. 5026, are ready and will be distributed at the conference. More info is available at <http://www.springeronline.com/978-3-540-68621-7>.
- Registration fees are very reasonable and the registration can be done on-line (preferred) or by faxing a filled-out form to the conference secretariat. For all details, see <http://www.math.unipd.it/ae2008/re.html>. Don't delay!
- For the latest information consult the conference web site.

Jun 20 — SC22/WG9 meeting

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Subject: Review of Ada Issues for June 2008 SC22/WG9 meeting (fwd)
To: Ada-Belgium mailing list <ada-belgium-info@cs.kuleuven.be>
Date: Fri, 28 Mar 2008 00:15:21 +0100 (MET)

As you may know, there is an upcoming meeting of ISO's Ada language working group (ISO/IEC JTC1/SC22/WG9) scheduled at the end of the Ada-Europe 2008 conference next June in Venice, Italy.

The Chairman of the Ada Rapporteur Group (ARG) of WG9 informed the Heads of Delegation that the Ada Issues (AIs) listed below have entered Editorial Review, and are intended to be submitted to WG9 for approval at the above mentioned meeting.

The AIs can be found at <http://www.ada-auth.org/AI05-SUMMARY.HTML>.

AI05-0004-1/11 2008-02-21 — Presentation issues in the Standard

AI05-0006-1/03 2007-11-29 — Nominal subtypes for all names

AI05-0013-1/11 2008-02-21 — No_Nested_Finalization is difficult to enforce

AI05-0022-1/03 2007-11-20 — Container tampering should be checked for formal subprograms

AI05-0023-1/05 2008-02-21 — 'Read on records with variant parts

AI05-0026-1/03 2007-11-20 — Missing rules for Unchecked_Unions

AI05-0027-1/03 2007-11-20 — Behavior of containers operations when passed finalized container objects

AI05-0029-1/03 2008-02-08 — Meaning of 12.5(8)

AI05-0030-2/03 2008-02-25 — Requeue on synchronized interfaces

AI05-0032-1/02 2007-11-11 — Extended return statements for class-wide functions

AI05-0033-1/02 2007-11-09 — Rules for non-library level interrupt handlers

AI05-0034-1/04 2007-11-26 — Categorization of limited views

AI05-0036-1/01 2007-01-18 — Number of characters to be output for Text_IO for enumerations

AI05-0038-1/03 2007-11-26 — Minor Errors in Ada.Text_IO.

AI05-0039-1/02 2007-11-27 — User-defined stream attributes cannot be dynamic

AI05-0041-1/06 2008-02-22 — Can a derived type be a partial view?

AI05-0042-1/02 2007-11-28 — Overriding versus implemented-by

AI05-0044-1/03 2007-11-29 — Equivalence and equality in containers

AI05-0045-1/04 2007-11-28 — Termination of unactivated tasks

AI05-0047-1/05 2008-02-26 — Annoyances in the array packages

AI05-0048-1/02 2007-11-29 — Redispatching is not expected in language-defined subprograms

AI05-0052-1/05 2008-02-24 — Coextensions and distributed overhead

AI05-0053-1/04 2008-02-25 — Aliased views of unaliased objects

AI05-0058-1/01 2007-08-01 — Abnormal completion of an extended return statement

AI05-0060-1/05 2008-02-26 — The definition of Remote access types is too limiting

AI05-0062-1/02 2007-11-29 — Null exclusions and deferred constants

AI05-0063-1/03 2008-02-24 — Access discriminants on derived formal types

AI05-0064-1/01 2007-09-13 — Redundant finalization rule

AI05-0065-1/02 2007-12-03 — Remote access types should be defined as externally streamable

AI05-0066-1/04 2008-02-22 — Temporary objects are required to live too long

AI05-0068-1/02 2007-11-29 — Inherited subprograms may be both abstract and requires overriding

AI05-0070-1/01 2007-10-24 — Elaboration of interface types

AI05-0072-1/01 2007-10-24 — Termination only signals 'Terminated when it is True

AI05-0073-1/03 2007-12-03 — Questions about functions returning abstract types

AI05-0076-1/02 2008-02-25 — Meaning of "function with a controlling result"

AI05-0077-1/02 2008-02-25 — The scope of a declaration does not include any context_clause

AI05-0078-1/02 2008-02-22 — Alignment need not match for Unchecked_Conversion

AI05-0079-1/02 2008-02-26 — An other_format character should be allowed wherever a separator is allowed

AI05-0080-1/02 2008-02-22 — "view of" is not needed when it is clear from context

AI05-0082-1/02 2008-02-22 — Accessibility level of generic formal types

AI05-0084-1/02 2008-02-26 — Pragma Remote_Types for Container library units

AI05-0086-1/01 2008-01-28 — Statically compatible needs to take null exclusions into account

AI05-0087-1/02 2008-02-22 — Formal nonlimited derived types should not have limited actual types

AI05-0088-1/01 2008-01-30 — Only the value of "**" is equivalent to repeated "*"s

Those AIs are now being circulated within the Ada community for review, with the intention to return comments to the ARG in time to properly answer them before the WG9 meeting.

Comments for the Belgian delegation should be sent to me at <Dirk.Craeynest@cs.kuleuven.be>. The deadline is 18:00 GMT+2, Tuesday, May 13th, 2008. Early comments are encouraged.

Dirk Craeynest

ISO/IEC JTC1/SC22/WG9, Head of Delegation, Belgium

Dirk.Craeynest@cs.kuleuven.be (for Ada-Belgium/-Europe/SIGAda/WG9 mail)

[See also "Ada-Belgium — Upcoming SC22/WG9 meeting" in AUJ 28-2 (Jun 2007), pp.69–70. —su]

Oct 15 — SPARK User Group 2008

From: Rod Chapman <rod.chapman@praxis-cs.co.uk>
Subject: SPARK User Group 2008
Date: Mon, 28 Apr 2008 09:02:07 −0700 (PDT)
Newsgroups: comp.lang.ada

We're pleased to announce that the next SPARK User Group meeting will be held in Bath, UK, on Wednesday 15th October 2008.

Booking a place

Capacity is limited, so priority will be given to our supported customers, other clients, tool partners and academic users. If you have not already received an invitation, then please contact us via sparkinfo@praxis-his.com.

Programme

The provisional list of speakers includes:

Guest speaker: Duncan Brown (Rolls-Royce plc).

"Formal Methods and DO-178C"

Industrial case-study: Neil White (iFACTS Team, Praxis).

"The iFACTS project"

Research report: Dr Paul Jackson, (University of Edinburgh)

"Using SMT Solvers to Prove SPARK VCs"

R&D Report: Rod Chapman (Praxis)

"SPARK Update and Release 7.6 Highlights"

[See also "SPARK-related events" in AUJ 28-3 (Sep 2007), p.134. —su]

Oct 26–30 — SIGAda 2008

From: Michael Feldman <mfeldman@gwu.edu>
Date: Fri, 09 May 2008 12:26:01 −0700
Subject: SIGAda 2008 Conference Submission Deadline EXTENDED to May 31, 2008
Newsgroups: comp.lang.ada

I just want to take a minute of your time to let you know the submission deadline has been extended to May 31 for the ACM SIGAda 2008 International Conference, which will take place in Portland, Oregon, Oct. 26–30, 2008.

The Call for Participation is on the web at

http://www.acm.org/sigada/conf/sigada2008/

We are soliciting Technical Articles, Extended Abstracts, Experience Reports, Workshops, Panel Sessions, and Tutorials.

Contributions from practitioners, researchers, students and faculty are actively solicited. Note that educator grants are available to cover conference and tutorial registration.

Thanks for your time; I hope to see you at the conference!

Michael Feldman

The George Washington University (emeritus)

Conference Chair, SIGAda 2008

Ada and Education

New Ada 2005 textbook

From: John McCormick <mccormick@cs.uni.edu>
Date: Mon, 17 Mar 2008 06:16:59 −0700 (PDT)
Subject: Re: Textbooks
Newsgroups: comp.lang.ada

For novice programmers have a look at my CS1 and CS2 books:

- "Programming and Problem Solving with Ada" by Dale, Weems, and McCormick. Jones and Bartlett, 2000. ISBN 0-7637-0792-9.
- "Ada Plus Data Structures" by Dale and McCormick. Jones and Bartlett, 2007. ISBN 0-7637-3794-1.

Webminar: GPS InSight

From: AdaCore Developer Center
Date: Tuesday March 11, 2008
Subject: GNAT Pro 6.1 InSight webinar
RSS: http://www.adacore.com/2008/03/11/gnat-pro-611-insight-webinar/

Tuesday, May 6, 2008: 8:00 am PST / 11:00 am EST / 17:00 CET

Summary

The latest version of the GNAT Pro Ada toolset sees over 150 enhancements to the technology including:

- Additional GNAT Pro platforms incorporating the gcc 4.1 code generator (this code generator will now be included on most platforms)
- Upgrade of the debugging engine, based on gdb 6.6
- Improvement in robustness and efficiency for Ada 2005 features
- Better real-time support on win32 platforms
- Thread-safe profiling with gprof, on several platforms
- Increased coverage analysis support for Ada in the gcov tool
- New warnings to help programmers detect errors earlier
- GNAT Pro companion tools such as gnatcheck, gnatpp and gnatmetric are being enhanced to support a wider variety of coding styles and coding standards.

The next webinar in the GNAT Pro InSight series will describe and demo some of the new features introduced in 6.1. As always, we will allow a question and answer session at the end enabling you to talk directly with the designers of GNAT Pro.

Greg Gicca and Cyrille Comar will present this webinar and answer your questions. To register for this webinar, please [go to https://adacore.webex.com/adacore/onstage/g.php?t=a&d=719427150 —su]

From: AdaCore Developer Center
Date: Monday May 12, 2008
Subject: GNAT Pro 6.1 InSight webinar archive
RSS: http://www.adacore.com/2008/05/12/gnat-pro-61-insight-webinar-archive/

The recently held GNAT Pro InSight webinar featuring GNAT Pro 6.1 is now available for viewing at:

www.adacore.com/home/gnatpro/webinars

[See also same topic in AUJ 29-1 (Mar 2008), p.7. —su]

Ada-related Resources

Ada sub-Reddit

From: Marc A. Criley <mc@mckae.com>
Date: Thu, 29 May 2008 19:51:44 −0500
Subject: Ada sub-Reddit created
Newsgroups: comp.lang.ada

An Ada programming language specific sub-Reddit has been created at:

http://reddit.com/r/ada

Since mentioning "Ada" in a Programming Reddit submission almost (but not always) guarantees downvotes, it's hoped an Ada-specific sub-reddit will maintain a bit more visibility for items of interest.

The sub-reddit is only a day old, so submissions are a little lean so far :-)

[See also "Ada Social Networks" in AUJ 29-1 (Mar 2008), pp.7–8. —su]

Ada-related Tools

Fuzzy sets for Ada

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Some updates
Date: Sun, 30 Mar 2008 19:13:17 +0200
Newsgroups: comp.lang.ada

Tables 1.8

http://www.dmitry-kazakov.de/ada/tables.htm

Strings Edit 2.1

http://www.dmitry-kazakov.de/ada/strings_edit.htm

Simple components 2.8

http://www.dmitry-kazakov.de/ada/components.htm

Fuzzy sets for Ada 5.2

http://www.dmitry-kazakov.de/ada/fuzzy.htm

The focus of these releases is UTF-8 support. New features include

Tables of case-insensitive tokes, UTF-8 encoded. Tables can be matched against an UTF-8 encoded strings in order to get the longest alternative. Chains of blank code points can be considered equivalent. Code points can be ignored (hyphens, for example). Case equivalence is determined according to the Unicode categories.

A fully functional UTF-8 substitute for Ada.Strings.Maps is provided. The implementation is enhanced in order to support sets defined by an indicator function additionally to a set of ranges.

An equivalent to Ada.Strings.Maps.Constants is provided as well.

Unicode code points characterization into all categories defined by the standard.

Blocks of Unicode characters as defined in the standard.

The license is as always GM GPL.

[See also same topic in AUJ 29-1 (Mar 2008), p.206. —su]

Simple components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: Simple components for Ada v3.0
Date: Sun, 11 May 2008 16:42:00 +0200
Newsgroups: comp.lang.ada

This version provides implementations of various locking primitives:

- Plain events which can be signaled, reset and awaited for;
- Arrays of events which can be signaled, reset and awaited in any combination of in race condition free way;
- Arrays of reentrant mutexes, deadlock-free with an enforced order of locking;
- Race condition free event distributing pulsed value to multiple tasks;
- Reentrant mutexes;
- Race condition free pulse events.

The documentation provides a discussion of using protected objects in Ada. It represents a programming pattern for using entry parameter values in the barrier and shows a way of avoiding race conditions. It also includes

- A solution of the problem of tasks synchronization at a check point based on sets of events;
- A solution of the dining philosophers problem based on sets of mutexes.

http://www.dmitry-kazakov.de/ada/components.htm

[See also same topic in AUJ 29-1 (Mar 2008), p.8. —su]

From: Ada Programming Blog
Author: Dmitry A. Kazakov
Date: Monday, May 12, 2008
Subject: Concurrency in Ada
RSS: http://ada-programming.blogspot.com/2008/05/ concurrency-in-ada.html

Ada is renown for its concurrency support. Parallel programming is a difficult issue in all aspects. It is difficult to learn, to design, to program, to validate, but for all, it is most difficult to reuse.

Yesterday I finished the version 3.0 of the Simple Components for Ada, where I

tried to summarize my experience and ideas in this area. The library among other things contains some basic gears for dealing with concurrency.

The section 9 is devoted to implementation of some lock-free data structures. These become more popular with new multi-core architectures. Though Ada was not designed to provide lock-free primitives on the low-level, for that obvious reason that this would be non-portable. It still has necessary tools. Here I mean the pragma Atomic, which allows many interesting things to do.

The section 10 contains implementations of locking synchronization primitives. Protected objects introduced in Ada 95 is an excellent mechanism of a great power. Especially interesting is to explore the requeue statement. Events, pulse events, arrays of events, mutexes, arrays of mutexes let be implemented as protected objects. Using the requeue statement one can do a lot of things, which appear impossible at the first glance. For example, this section presents a programming pattern for using entry parameters in the barriers of. It also discusses how race condition and deadlocks can be avoided when using protected objects.

Two classical problems are considered on examples: the checkpoint synchronization problem, and the dining philosophers one.

It is free as only it can be, the license is GMGPL I hope you will enjoy it.

http://www.dmitry-kazakov.de/ada/components.htm

Ada support in LLVM

From: Duncan Sands <duncan.sands@math.u-psud.fr>
Subject: Announcement: GNAT ported to LLVM
Date: Sun, 23 Mar 2008 15:05:59 −0700 (PDT)
Newsgroups: comp.lang.ada

This is to let people know that the recently released LLVM 2.2 compiler toolkit contains experimental support for Ada through the llvm-gcc-4.2 compiler. Currently the only platform it works on is linux running on 32-bit Intel x86. This is because that's what I run, and I'm the only one who's been working on this. I would appreciate help from other Ada guys, both for porting to new platforms and adding support for missing features, not to mention testing and bug fixing!

LLVM (http://llvm.org/) is a set of compiler libraries and tools for optimization and static and just-in-time code generation. Personally I find LLVM a lot of fun, and pleasant to work with due to the good design and clean implementation. I hope you will too! llvm-gcc is gcc with the gcc optimizers replaced by LLVM's; llvm-gcc-4.2 is the version of llvm-gcc based on gcc-4.2.

The way llvm-gcc works (this is transparent to users) is that the gcc-4.2 GNAT front-end converts Ada into "gimple", gcc's internal language independent representation. The gimple is then turned into LLVM's internal form, referred to as IR. This in then run through LLVM's optimizers, followed by LLVM's code generators which squirt it out as assembler or object code. In practice you can use llvm-gcc as a drop in replacement for gcc. However the use of LLVM opens up other possibilities too.

For example, it is possible to have llvm-gcc squirt out LLVM IR rather than object code (by using -emit-llvm on the command line). It is possible to link the LLVM IR for different compilation units together and reoptimize them. In other words you can do link-time optimization. This is all language independent, so if part of your program is written in Ada and other parts in C/C++/Fortran etc, you can link them all together and mutually optimize them, resulting in C routines being inlined into Ada etc.

The compiler works quite well, but it is still experimental. All of the ACATS testsuite passes except for c380004 and c393010. Since c380004 also fails with gcc-4.2, that makes c393010 the only failure due to the use of the LLVM infrastructure (the problem comes from the GNAT front-end which produces a bogus type declaration that the gimple -> LLVM convertor rejects). On the other hand, many of the tests in the GNAT testsuite fail. The release notes give some more details of what works and what doesn't:

http://llvm.org/releases/2.2/docs/ReleaseNotes.html

The precompiled llvm-gcc-4.2 shipped with the LLVM 2.2 release was built without support for Ada, so you will need to build the compiler yourself. You can find instructions at

http://llvm.org/docs/GCCFEBuildInstrs.html

Please report bugs and problems to the LLVM mailing lists, or using http://llvm.org/bugs/ One nice thing about LLVM is that people are responsive and quickly fix bugs (often by the next day).

The LLVM IR is easy to read (with a bit of practice), and since it contains the entire LLVM state you get to see exactly what has happened to your program. This might be useful for static analysis, it is certainly useful for understanding how the various Ada constructs are implemented. To give you a taste for what it looks like, here is an example showing what a simple Ada program gets turned into. [See original post for a complete example of code generation —su]

I hope you have fun playing with LLVM!

[See also "LLVM Compiler Infrastructure" in AUJ 29-1 (Mar 2008), p.75. —su]

From: Samuel Tardieu <sam@rfc1149.net>
Subject: Re: Announcement: GNAT ported to LLVM
Date: Mon, 24 Mar 2008 10:25:16 +0100
Newsgroups: comp.lang.ada

Thanks Duncan, this is an outstanding contribution to the Ada community. Given that LLVM is already ahead of GCC in terms of code generation quality (sometimes, starting from zero and choosing another path is a competitive advantage), this looks very promising.

Note that the GCC folks are happy with LLVM adding some competition in the open-source compilers arena. This is very motivating.

The difficult task, as you already know, will be to keep the Ada front-ends in both compilers in sync. I wish you good luck with that!

From: Duncan Sands <duncan.sands@math.u-psud.fr>
Subject: Re: Announcement: GNAT ported to LLVM
Date: Mon, 24 Mar 2008 11:09:26 −0700 (PDT)
Newsgroups: comp.lang.ada

I'm glad you appreciate my work! That said, in my experience gcc-4.2 produces slightly faster code for Ada than llvm-gcc-4.2 does. Given that LLVM manages to produce code that comes close to gcc while being much simpler than gcc and easier to improve, I expect it will overtake GCC soon. In fact I haven't even started working on Ada specific optimizer improvements yet: I've been concentrating on correctness.

It's not yet clear to me whether I should backport the gcc-4.3 Ada front-end to llvm-gcc-4.2, or start working on llvm-gcc-4.3. For the moment I'm just working on improving the correctness and robustness of llvm-gcc-4.2.

From: Duncan Sands <duncan.sands@math.u-psud.fr>
Subject: Re: Announcement: GNAT ported to LLVM
Date: Thu, 27 Mar 2008 01:27:55 −0700 (PDT)
Newsgroups: comp.lang.ada

> I assume that due to the link-time optimization capability that inlining among packages will be handled naturally.

that's correct: you can compile each package to bitcode, then at link-time they will all be mutually optimized, including inlining into each other. You can also compile the entire runtime to bitcode and have that be mutually optimized with your code too. I didn't turn this on by default because currently link-time-optimization

is not transparent: you have to explicitly call some LLVM tools at link time. There's a plan to teach llvm-gcc to do this automagically when you use it to do linking.

> GNAT-gcc can't do that, right?

It can to some extent: if you use -O2 -gnatn then it will peek inside the bodies of packages you are using to try to inline functions. That functionality becomes a lot less useful now though.

> This alone ought to be a big deal as accessor/setter conventions are leading to programs filled with tiny procedures and functions.

Very true, and I guess that's why ACT implemented -gnatn.

JGNAT is coming back

From: Kickin' the Darkness
Date: Thursday, May 22, 2008
Subject: JGNAT is coming back!
RSS: http://blog.kickin-the-darkness.com/2008/05/jgnat-is-coming-back.html

AdaCore has announced in their latest newsletter that JGNAT, their Ada-to-Java Byte Code compiler, is being updated and will be made available in the 2nd quarter of 2008 (see the "In The Pipeline" sidebar):

A collection of add-on tools for interfacing between Ada and Java is scheduled for release during Q2 2008. They support mixed-language Ada/Java development, in particular:

- Calling natively-compiled Ada code from Java
- Compiling Ada to JVM bytecodes and communicating between Ada and Java directly.

The toolsuite exploits the Java Native Interface (JNI) for the first scenario, but automates the generation of the JNI-related "glue code" to ease the job of the developer. An updated version of AdaCore's JGNAT product handles the second scenario. The tools take advantage of Ada 2005's new features to provide an interfacing mechanism that complies with the Ada standard. A future version of the toolsuite will support the invocation of Java methods from natively-compiled Ada code.

The original release was never updated beyond version 1.1p, which I used a fair amount. I felt it was almost, but not quite, production quality. My experience was that it worked pretty well with Java 1.2, tasking broke in 1.3, and was pretty much a non-starter for 1.4.

Robert Dewar, AdaCore's president, commented in 2004 that "the status of JGNAT is that we have kept the sources updated to the minimal extent that they compile, but we no longer support this product and it was never fully completed."

I don't know the impetus behind restarting JGNAT support, but I'm glad to see it happening.

[See also "JGNAT and MGNAT" in AUJ 27-3 (Sep 2006), pp.150–151. —su]

VAD 7.1 — Visual Ada Developer

From: Leonid Dulman <leonid_dulman@yahoo.co.uk>
Newsgroups: comp.lang.ada
Subject: Announce: Visual Ada Developer (VAD) version 7.1
Date: Sun, 16 Mar 2008 18:08:15 −0000

Visual Ada Developer VAD

VAD is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. VAD is distributed in the hope, that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

VAD 7.1 Common description.

1. VAD (Visual Ada Developer) is a Tcl/Tk oriented Ada-95(TCL) GUI builder portable to difference platforms, such as Windows NT/9x,Unix (Linux), Mac and OS/2. You may use it as IDE for any Ada 95(05) (C,C++,TCL) project.

You may use it to build TCL script only. VAD generated ada sources,you may compile and build executable with GNAT in Windows and Unix(Linux) or Aonix ObjectAda 7.2 in Windows.

[...]

VAD 7.1 is available in http://www.websamba.com/guibuilder You may dowload sources vad71scr.tar.bz2, vadhlp.tar.bz2, vadtutor.tar.bz2, vadsmp.tar.bz2,vad71aonix.tar.bz2,adahlp .tar.bz2, vadtcl.tar.bz2, adastyle.tar.bz2,philosofers.tar.bz2,vadidl. tar.bz2

and binaries vad71win.tar.bz2 (Windows 9x/NT) vad71lin.tar.bz2 (i386) Any questions, any ideas, any problems, any help

[See also "VAD 6.3 — Visual Ada Developer" in AUJ 27-4 (Dec 2006), p.200. —su]

QtAda binding

From: Vadim Godunko <vgodunko@gmail.com>
Subject: Announce: QtAda 1.0.3 released
Date: Wed, 7 May 2008 08:43:47 −0700 (PDT)
Newsgroups: comp.lang.ada

We are pleased to announce QtAda 1.0.3 release. This release includes bugs fixes and enhancements. You can see the full list at the end of this mail.

QtAda is an Ada 2005 language bindings to the Qt libraries and a set of useful tools. QtAda allows easily to create cross-platform powerful graphical user interface completely on Ada 2005. QtAda applications will work on most popular platforms — Microsoft Windows, Mac OS X, Linux/Unix — without any changes and platform specific code. QtAda allows to use all power of visual GUI development with Qt Designer on all software lifecycle stages — from prototyping and up to maintenance. QtAda is not just a bindings to the existent Qt widgets, it also allows to develop your own widgets and integrates it into the Qt Designer for high speed visual GUI development.

Multi platform source code package and Microsoft Windows binary package of the QtAda 1.0.3 can be downloaded from:

http://www.qtada.com/

[...]

From: Vadim Godunko <vgodunko@gmail.com>
Subject: Re: Announce: QtAda 1.0.3 released
Date: Mon, 12 May 2008 06:00:47 −0700 (PDT)
Newsgroups: comp.lang.ada

> Is it for Qt3 or Qt4?

Qt 4.3 and later.

[See also same topic in AUJ 29-1 (Mar 2008), p.10. —su]

GNATGPR — GPR project files

From: David Sauvage <sauvage.david@gmail.com>
Subject: Announce : Release of gnatgpr, access to GPR project information.
Date: Mon, 24 Mar 2008 14:52:06 −0700 (PDT)
Newsgroups: comp.lang.ada

GNATGPR is an Ada 2005 GPL software, it allows the user to do simple information request on GNAT GPR project files, like for example:

- Give all included projects.
- Give all included sources.
- Give all main files.
- Give all included object paths.

There are 2 ways of accessing those services:

- In the Shell, using the gnatgpr binary.
- In Ada, using the GNAT_GPR package specification interface.

GNATGPR is based on:

- GNAT GPL 2007 (GPL), re-use of the Ada front-end. One of the main changes is that it got his own Namet and so on.
- AdaControl (GMGPL), re-use of the Options_Analyzer.
- AUnit (GPL), for unit testing.

Thanks to all the people behind those projects.

https://gna.org/projects/gnatgpr/

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Subject:Re: Annonce : Parution de gnatgpr, logiciel d'accès aux informations des projets GPR.
Date: Thu, 20 Mar 2008 04:06:21 −0700 (PDT)
Newsgroups: fr.comp.lang.ada

[Translated from French —su]

It would be nice to incorporate these functions in gnatls.

> For this to be incorporated, we must make an assignment of copyright, etc…, while a separate tool does not have these constraints.

Of course, but the benefits largely offset this:

- more users
- a better chance that the program is maintained on the long-term
- a better chance that the program is compatible with future versions of GCC
- having his name in the list of GCC contributors
- do not need a new Debian package :)

The separate tool can of course be used to test new functions before they are incorporated.

AdaDesigner — Refactoring framework

From: David Sauvage <sauvage.david@gmail.com>
Date: Mon, 26 May 2008 05:27:12 −0700 (PDT)
Subject: Announce : Release of AdaDesigner, to an Ada refactoring framework.
Newsgroups: comp.lang.ada

AdaDesigner [1] is an Ada 2005 (GPL) software which aims to become an Ada software refactoring framework.

Using ASIS [2], AdaDesigner builds a specific package call-graph view of the project. Using some meta-data added in the GPR project files, it creates an abstraction layer further called sub-component layer. One sub- component contains several packages.

AdaDesigner already proposes software engineers one interesting component design view of their software projects. This is done by the generation of a graphviz [3] DOT [4] file.

See AdaDesigner presentation [5] for more details.

In the future, AdaDesigner could also be able :

- To generate UML components views directly from the code (via XMI [6] format).
- To propose a modern way of doing refactoring on the software (via a high level HMI).

AdaDesigner is based on :

- GNATGPR [7] (GPL)
- ASIS [8] (GPL)
- AUnit [9] (GPL)

Thanks to all the people behind those projects.

https://gna.org/projects/adadesigner

[1] https://gna.org/projects/adadesigner

[2] http://www.sigada.org/WG/asiswg/

[3] http://www.graphviz.org/

[4] http://www.graphviz.org/doc/info/lang.html

[5] http://home.gna.org/adadesigner/presentation_adadesigner.pdf

[6] http://en.wikipedia.org/wiki/XMI

[7] https://gna.org/projects/gnatgpr

[8] https://libre.adacore.com/

[9] https://libre.adacore.com/aunit

Hibachi binaries

From: Tom Grosman <grosman@aonix.fr>
Date: Fri, 25 Apr 2008 07:13:29 −0400
To: hibachi-dev@eclipse.org
Subject: Hibachi binaries are now available

We have put up the first binary builds on the eclipse.org servers. To access them, go to the Hibachi homepage (www.eclipse.org/hibachi) and click on the Downloads link on the left menu. We have not created an update site on the Eclipse servers for this 'Integration" build.

Since this is the first integration build (the release will be in a month) of the plugin, there will likely be some interesting "features" that users encounter. I remind everyone that bugzilla is used to consult, update and create bugs, as well as submit bug fixes. So if you don't yet have and account, I strongly encourage you to create one at http://bugs.eclipse.org/bugs.

Documentation exists in the plugin, more is on the way. I encourage you to contribute, including to the Hibachi wiki, also accesible via the project homepage.

Vision2Pixels

From: Pascal Obry <pascal@obry.net>
Date: Mon, 03 Mar 2008 17:09:42 +0100
Subject: ANNOUNCE — Vision2Pixels
Newsgroups: comp.lang.ada

Vision2Pixels

http://v2p.fr.eu.org/

Pascal Obry and Olivier Ramonat are pleased to announce the launch of Vision2Pixels Web application. This application is oriented toward photography critics.

As photographers we used another Web application some time ago, the maintainers have decided to close it, and as the code was not Open Source, nobody was able to take the maintenance back. So we decided two years ago to launch this project and to start for good as a GPL project. At least this project won't depend on our will or energy to continue it or not.

The Web interface is in French and oriented toward French critics only.

This project is built in Ada (hence the post in this forum) and is using AWS/Ajax support (the new AWS's Web_Block Ajax support is born while building Vision2Pixels as the templates2ada tool) and Gwiad dynamic plug-in project from Olivier.

We just put out the first release, please bear with us, it is not perfect and is missing many features (like searching). We plan to continue working on this project of course, even if the goal is more to used it to post our own pictures.

We take the opportunity to call for contributions. We have no designer on the project and needs help on the core Ada too. The code comments and issue tracker are all in English.

In any case this is a good show-case for Ada and Ajax.

It builds fine with latest GNAT GPL 2007.

During the project we have switched from Subversion to Mercurial then to Git. Git is really wonderful but this is another story. [...]

The issue tracker:

http://code.google.com/p/vision2pixels/issues/list

Thanks to everyone for the help received, notably to host the project. And thanks also to AdaCore for the GNAT GPL compiler.

Canta

From: Christophe Chaumet <christophe.chaumet@chaumetsoftware.com>
Date: Wed, 12 Mar 2008 19:56:59 +0100
Subject: Info: un logiciel grand public sous Windows entièrement en Ada
Newsgroups: fr.comp.lang.ada

[Translated from French —su]

Lady Ada Lovelace is pleased to announce the birth of Canta, a beautiful 303 Kb software; the mother and baby are doing well.

100% Ada, it runs on Windows XP and Vista, thanks to the win32ada binding and the GNAT GPL compiler.

It is a software for the large public intended to help sing, and this concerns a lot of people, doesn't it?

A downloadable demo version is available at this address:

http://www.chaumetsoftware.com/

Ada-related Products

AdaCore — GNATbench 2.1.0

From: AdaCore Press Center
Subject: AdaCore Announces the Release of GNATbench 2.1.0
Date: Tuesday April 15, 2008
RSS: http://www.adacore.com/2008/04/15/gnatbench210/

Enhanced Eclipse-based development environment for Ada improves productivity; decreases time-to-market

NEW YORK and SAN JOSE, Calif., April 15, 2008 — Embedded Systems Conference, Silicon Valley — AdaCore, provider of the highest quality Ada tools and support services, today announced a new major release of GNATbench, the company's Eclipse-based development environment for Ada. GNATbench 2.1.0 includes a variety of enhancements, including general project management and presentation capabilities, new features within the language-sensitive editor, new source navigation capabilities, new wizards, and an enhanced builder. The result is a more powerful Integrated Development Environment (IDE) for Ada that supports tighter integration with Wind River's Workbench development suite, and automatic integration with the large pool of software development capabilities already available within the Eclipse framework.

"This latest GNATbench release further enhances an already powerful set of software development capabilities," said Pat Rogers, AdaCore's GNATbench Project Leader. "We at AdaCore believe that customer support is our primary responsibility. This release adds a variety of new capabilities, many based on user input, to improve developer productivity and decrease time-to-market."

"AdaCore was founded on open source software principles to provide the highest quality software development solutions and support for the Ada programming language," said Robert Dewar, President and CEO of AdaCore. "We joined the Eclipse Foundation a year ago as an Add-In-Provider and Member-At-Large so that we could contribute to this open source community. The Eclipse open source framework allows us to extend our core principles to support the largest number of integrated third-party tools available for Ada software developers."

"Wind River is pleased to see AdaCore's new release of the GNATbench product," states Rob Hoffman, general manager of aerospace and defense, Wind River. "GNATbench has always been seamlessly integrated with Wind River Workbench, and this latest release further advances the Ada capabilities available to the Workbench software developer for both all-Ada and mixed-language applications."

GNATbench 2.1.0 specifically enhances general Project Management and Presentation capabilities, including:

- Independent project hierarchies
- The ability to clean project hierarchies
- And the ability to fully restore projects form configuration management systems

For the Language-Sensitive editor, some new features include:

- A light bulb with quick fix suggestions, next to source errors
- Smart space key, for abbreviation expansions
- Automatic construct closing insertion
- Smart tab for logical indenting
- Smart formatting
- Smart comment wrapping
- Special color coding for annotation comments
- Standard parenthesis highlighting
- Code assist for recently chosen completions

New Source Navigation capabilities include:

- Enhanced open declaration or body action
- Next or previous subprogram entry navigation

The new Wizards include:

- Create new Ada source folder or subfolder
- Create new Ada source file with specific headers
- Create new Ada project with advanced project settings

And finally, the Builder enhancements are:

- Automatic file save before building
- Automatic console display for the builder
- Scenario settings are now persistent across sessions
- New builder fast action key settings
- Tool chain selection from within the IDE
- Linker messages are now integrated within the problem view window

All these enhancements are added on top of the already powerful features that were provided in the three previous releases of GNATbench.

About AdaCore

Founded in 1994, AdaCore is the leading provider of commercial software solutions for Ada, a modern programming language designed for large, long-lived applications where safety, security, and reliability are critical. AdaCore's flagship product is the GNAT Pro development environment, which comes with expert on-line support and is available on more platforms than any other Ada technology. AdaCore has an extensive worldwide customer base; see http://www.adacore.com/home/company/customers/ for further information.

Ada and GNAT Pro continue to see growing usage in high-integrity and safety-certified applications, including commercial aircraft avionics, military systems, air traffic management/control, railroad systems, and medical devices, and in security-sensitive domains, such as financial services.

[See also "AdaCore — GNATbench Eclipse Plug-in" in AUJ 28-2 (Jun 2007), pp.77–78. —su]

From: AdaCore Press Center
Subject: GNATbench 2.1.0
Date: March 7, 2008
RSS: http://www.adacore.com/2008/03/07/gnatbench-210/

GNATbench 2.1.0 supports Eclipse 3.3, with version 4.0 of the C/C++ Development Tools (CDT), on the Linux (x86 and x86-64), Solaris (SPARC), and Windows platforms.

GNATbench 2.1.0 introduces the following features among others:

Project Management and Presentation

- Independent Project Hierarchies
- Cleaning Project Hierarchies
- Fully Restorable Projects
- Problems View Entries for GNAT Project Files

Language-Sensitive Editor Enhancements

- "Quick Fix" for Ada
- Smart Space Key
- Automatic Construct Closing
- Smart Tab Key
- Smart Enter Key
- Text and Comment Lines Refilled Against Margin
- Special Coloring for "Annotation Comments"
- The "Show In" Contextual Menu Entry Supported
- Improved Comment Block Selection
- Standard Parenthesis Highlighting

- Integration of Code Formatting Features
- Recent Chosen Completions Displayed at the Top

Additional Wizards

- A "New Ada Source Folder" Wizard
- A "New Ada Source File" Wizard
- Additonal "New Ada Project Wizard"

Builder Enhancements

- Altered Ada Files Saved Automatically
- Console View Visible Automatically
- Persistent Scenario Variable Settings
- Builder Command Key Bindings
- Compiling Individual Files via the Ada Project Explorer
- Toolchain Selection
- Linker Messages In the Problems View

Source Code Navigation Enhancements

- Enhanced Open Declaration / Open Body Actions
- Next / Previous Subprogram and Entry Navigation

Miscellaneous Improvements

- Ada Project Explorer Support for Double-Click Actions
- New "Ada" Category for GNAT Import Wizard
- Additional Icon Decorations in Ada Project Explorer

AdaCore — GNAT Pro for VxWorks 653

From: AdaCore Press Center
Subject: AdaCore Announces Support for VxWorks 653, Version 2.2
Date: Tuesday April 15, 2008
RSS: http://www.adacore.com/2008/04/15/do-178b/

Now Supporting Full Integration with VxWorks 653 Platform, Version 2.2

NEW YORK and SAN JOSE, Calif., April 15, 2008 — Embedded Systems Conference, Silicon Valley — AdaCore, provider of the highest quality Ada tools and support services, today announced the availability of GNAT Pro High-Integrity-Edition for DO-178B (version 6.1.1) on the latest version of the Wind River's VxWorks 653 Platform, Version 2.2 for avionics and mission-critical systems. The result is a uniquely powerful and seamless environment for the development of Ada and mixed-language software for safety-critical and other high reliability applications.

"AdaCore has enjoyed a long and successful relationship with Wind River, with GNAT Pro offering our customers the first Ada language development environment for the initial VxWorks 653 platform many years ago," said Robert Dewar, President and CEO of AdaCore. "By supporting this latest version of VxWorks 653, we are enhancing an already proven solution that has been certified to DO-178B Level A as part of multiple avionics systems, including the Boeing 787, C-130AMP cargo, and KC-767 tanker aircraft."

GNAT Pro High-Integrity Edition for DO-178B for VxWorks 653 implements a full ARINC-653 APEX API. With this latest release, developers can choose from four different run-time libraries, including:

- Full Ada run-time profile
- Zero-Foot-Print (ZFP) run-time profile, the smallest run-time library of the group, removing most dynamic features and providing the simplest certifiable solution
- Ravenscar-compliant run-time profile that adds tasking and some other dynamic features that have been proven to be certifiable
- Special Cert run-time profile, specifically designed to support the requirements of avionics systems

The product was specifically tailored to provide the features needed in the development of avionics systems, while removing unneeded features to simplify certification to DO-178B Level A.

All GNAT Pro High-Integrity Family members are also accompanied by AdaCore's GNATstack analysis tool set. GNATstack statically calculates the maximum stack space required by each task in an application. The computed bounds can be used to ensure that sufficient space is reserved, thus guaranteeing safe, predictable execution with respect to stack usage. GNATstack uses conservative analysis to deal with complexities such as subprogram recursion, while avoiding unnecessarily pessimistic estimates. The tool's output data can be used directly to satisfy DO-178B requirements (Table A-5, Objective 6, which relates to the Accuracy and consistency issues itemized in Section 6.3.4f).

In addition, GNAT Pro High-Integrity Edition for DO-178B is bundled with GNATbench 2.1.0, the latest release of AdaCore's powerful Eclipse-based development environment for Ada. GNATbench 2.1.0 tightens integration with Wind River's Workbench development suite, offering more advanced editing, automatic program traversal, new project wizards and more advanced build capabilities for both Ada-only and mixed-language application development.

"Wind River is pleased to see the GNAT Pro Ada development solution on our latest VxWorks 653 Platform," said Rob Hoffman, general manager of aerospace and defense, Wind River. "AdaCore and Wind River have a proven success record in supporting avionics system developers with our joint solutions. Now our customers can start taking advantage of the new capabilities and features available in our industry-leading VxWorks 653 Platform for both Ada and mixed language safety-critical development, using this latest version of GNAT Pro."

[See also "Aonix — ObjectAda RAVEN for VxWorks 653" in AUJ 28-4 (Dec 2007), pp.211–212. —su]

AdaCore — GPRbuild 1.1.0

From: AdaCore Press Center
Subject: GPRbuild 1.1.0
Date: April 24, 2008
RSS: http://www.adacore.com/2008/04/24/gprbuild-110/

AdaCore is pleased to announce the immediate availability of GPRbuild 1.1.0. This new release of GPRbuild offers full support for all major native and cross platforms supported by AdaCore (15 native configurations and 22 cross). It provides better support for projects that use a combination of languages (Ada, C, C++, Assembly, and others) and has enhanced its support of very large subsystems. New major features have also been added:

- Each subsystem can define which of its sources are visible to other subsystems. An error is reported when a dependency on a non-visible source of a different subsystem is detected.
- A new GPRbuild option limits source dependency to immediately "with"ed projects.

This new release has addressed all issues reported after the 6.1.1 GNAT Pro release.

We encourage all GNAT Pro users to upgrade to this new builder technology. As a reminder, using or upgrading GPRbuild does not require a change of compilers and therefore does not imply changing code generators. All the GNAT Pro compilers released over the last 5 years are in fact supported by GPRbuild.

GPRbuild 1.1.0 can be downloaded from the "Tools" folder in the "Download" section on GNAT Tracker. As always, for questions, or to inform us of issues that you encounter, please let us know through the GNAT Tracker report facility or by email to the usual report@adacore.com address.

[See also "AdaCore — GPRbuild" in AUJ 28-4 (Dec 2007), pp.210–211. —su]

Adalog — AdaControl

From: Jean-Pierre Rosen <rosen@adalog.fr>
Subject: AdaControl 1.9r4 released
Date: Mon, 28 Apr 2008 17:26:50 +0200
Organization: Adalog
Newsgroups: comp.lang.ada

Adalog is pleased to announce the release of version 1.9 of AdaControl. This version features 346 possible controls, and covers all of the checks from Gnatcheck (but in a more powerful and parameterizable way, of course ;-).

As usual, AdaControl is released under the GMGPL, and can be downloaded from http://www.adalog.fr/adacontrol2.htm

[See also same topic in AUJ 29-1 (Mar 2008), p.13. —su]

Excel Software — WinA&D 6.0

From: Excel Software News Desk
Subject: WinA&D 6.0 for Windows XP and Vista
Date: April 25, 2008
URL: http://www.excelsoftware.com/newswina&d600.html

System Models with Simulation, Software Design and Requirements Management

April 25, 2008 — Excel Software is pleased to announce WinA&D 6.0 for system models and simulation, requirements management, software design, code generation, reengineering and project reports. Version 6.0 has been optimized to run on all Windows XP and Vista computers. It includes hundreds of enhancements, new printed and PDF manuals, integrated help system and Vista friendly installer. Free videos can be downloaded from the company web site to learn about WinA&D.

Once installed, WinA&D can run from a standard user account. The tool supports many types of software design including UML for object-oriented design, structured analysis and design, data models for database systems and real-time, multi-task design. WinA&D is scalable to large projects, multiple users, distributed team development and the integration of thousands of documents across the development process.

A software designer can choose the best mix of software models and notations for their specific development project. WinA&D supports process, data, class, state, structure, object and task models with dozens of popular notations and hundreds of customization options. In addition to language independent models, source code can be generated from models or models created from code. The growing list of supported programming languages include C++, C#, Java, Delphi, PHP, Ada, SQL, C, Pascal, Basic and Fortran.

Requirement entries are defined as structured information in a Requirement Definition dialog controlled by a template with user-defined field and value choices. Requirement entries can be linked to any kind of target document, diagram or specific diagram objects to provide two-way traceability. Powerful features enable developers to easily link and navigate between thousands of requirement entries and targets. Requirement information is presented through user-defined views, queries and reports.

WinA&D has dozens of ready-to-run reports that present project diagrams, tables, specifications, dictionary and requirements to an HTML browser or word processor. The built-in scriptable report generator makes it easy to create custom reports with full control over content and format. The report generator has complete access to any data within any WinA&D document. Custom reports and user-scripted features can be included as menu commands or buttons in application dialogs.

WinA&D for Windows is available in a Standard edition for $495, Desktop edition for $1195 or Developer edition for $1995. Documents can be shared with MacA&D on Mac OS X. See the company web site for product information, site license pricing and secure online ordering.

[See also "ExcelSoftware — WinA&D & WinTranslator" in AUJ 25-2 (Jun 2006), pp.63–64. —su]

Excel Software — WinTranslator 3.1

From: Excel Software News Desk
Subject: WinTranslator 3.1 for Windows XP and Vista
Date: April 28, 2008
URL: http://www.excelsoftware.com/newswintranslator310.html

Generate Class Diagrams, Structure Charts and Data Models from Source Code

April 28, 2008 — Excel Software is pleased to announce immediate availability of WinTranslator 3.1 for Windows XP and Vista. The new edition includes enhancements, new printed and PDF manual, integrated help system and Vista friendly installer. Once installed, WinTranslator runs from a standard user account.

WinTranslator scans source code to extract data. That data is imported into WinA&D to automatically generate class diagrams, structure charts and data models. The project dictionary is populated with design details like data types, method arguments, descriptive comments and links from model objects to associated source code. Videos on the company web site show diagrams generated from code using WinTranslator and WinA&D.

WinTranslator is a highly scalable, fully automated tool that supports many programming languages and dialects. It generates models and dictionary information from legacy source code, class frameworks, open source projects and other reusable code assets. It provides a step-by-step process to quickly document an unfamiliar project.

WinA&D and WinTranslator work together to provide:

- UML Class Models from Object-Oriented C++, C#, PHP, Java, Delphi or Ada
- Rich Data Models with Indexes, Triggers, Primary and Foreign Keys from SQL Schema
- Structure Charts from Procedural C, Pascal, Basic, PHP or Fortran
- Multi-Level Diagrams with Objects Linked to Associated Source Code for Browsing
- Data Types, Arguments, Namespaces and Comments Captured from Code
- Project Scalability to Millions of Code Lines Across Thousands of Files and Folders
- Automated Structure Chart Generation from Source Code for Each Thread of Execution

To create models from code, the developer uses a step-by-step dialog in WinTranslator to select the programming language, code folders and other options. A script of commands is created and run to scan the code and output data to a text file. That file is imported into a new WinA&D project to populate the dictionary and generate a stack of diagrams that represent the code structure. The entire process takes just minutes to accurately document millions of lines of source code.

WinTranslator is $495 for a Single User License. It works in conjunction with the WinA&D, QuickUML or QuickCRC modeling tools. See the company web site for product information, site license pricing, demo edition, videos and secure online ordering.

[See also "ExcelSoftware — WinA&D & WinTranslator" in AUJ 25-2 (Jun 2006), pp.63–64. —su]

Rapita Systems — RapiTime 2.0

From: Rapita Systems
Date: 10 March, 2008
Subject: RapiTime 2.0 Improves the Timing Performance of Real-Time Embedded Software
RSS: http://www.rapitasystems.com/node/285

Rapita Systems Ltd., leaders in measurement based worst-case execution time analysis, announce RapiTime 2.0 a comprehensive toolset for performance profiling, and worst-case execution time analysis of real-time embedded software.

In the avionics, telecommunications, space, and automotive electronics industries, successful companies know the importance of understanding, verifying, and improving the timing performance of their real-time embedded software. Taking a systematic and scientific approach to ensuring that time constraints are met, allows smart engineers to build timing correctness into their systems rather than spend time and effort trying to get timing bugs out. The result is reduced time to market, combined with class-leading product reliability.

Using the new features of RapiTime 2.0 engineers can gain a clear, detailed, and accurate understanding of the execution time behaviour of their real-time software. RapiTime 2.0 uses an Eclipse-based graphical user interface to provide interactive access to a wealth of execution time data, obtained via on-target performance profiling and worst-case execution time analysis. For further details about the key information provided by RapiTime 2.0 download a copy of the new RapiTime Product Brochure and RapiTime White Paper.

Beyond providing an in-depth understanding of software timing behaviour, RapiTime 2.0 adds the capability to target optimisation effort precisely where it will have the maximum benefit. Using the advanced features of RapiTime 2.0, engineers can discover the extent to which different software components contribute to the worst-case (often very different from their contribution to the average case), identify worst-case hotspots, and select the best opportunities for source code optimisation.

RapiTime 2.0 also adds the capability to answer "what-if" questions about achievable performance gains, enabling the headroom for new functionality to be accurately assessed. Exploiting this advanced capability allows engineers to leverage their efforts, obtaining disproportionately large reductions in overall execution times through optimisations targeted at a tiny proportion of the overall software. The result is an elimination of timing overruns, and the efficient creation of headroom for new functionality, without the need for costly and time-consuming hardware upgrades. For further insight into how effective RapiTime is, download a copy of the RapiTime BAE Systems Experience Report.

RapiTime is a practical solution that can analyse complex embedded real-time software running on the latest high performance microprocessors. RapiTime 2.0 supports industrial scale C and Ada software from a few kBytes to millions of lines of code. Due to its simple, yet highly effective measurement-based approach, RapiTime 2.0 is compatible with virtually every 8, 16, and 32-bit embedded microprocessor on the market. RapiTime 2.0 supports both automatic software instrumentation and hardware assisted timing trace capture via a logic analyser, or dedicated hardware such as the RapiTime TraceBox.

Rapita Systems will be demonstrating the capabilities of RapiTime 2.0 at DATE08 (10th – 14th March 2008, Munich, Germany) on stand S3.

For more information about RapiTime on-target timing analysis solutions, contact Us.

[See also "Rapita Systems — RapiTime 1.3" in AUJ 29-1 (Mar 2008), pp.14–15. —su]

Ada and Microsoft

Ada for Microsoft WinCE

From: Ivan Levashew <octagram@bluebottle.com>
Subject: Re: Ada / MS WinCE
Date: Sun, 27 Apr 2008 02:57:32 +0700
Newsgroups: comp.lang.ada

> Is there any way to compile an Ada application for MS WinCE ?

The only way I have heard of is: GNAT for .NET => .NET CF => WinCE

From: Rob Veenker <veenker@xs4all.nl>
Subject: Re: Ada / MS WinCE
Date: Tue, 06 May 2008 20:02:20 +0200
Newsgroups: comp.lang.ada

AdaCore has already developed a GNATPro for .Net which generates true MSIL code. (GNAT Pro means it is a supported version; don't know about a possible public release) I noticed that they also got JGNAT running again. JGNAT is an Ada compiler for the Java runtime environment. We have actually used JGNAT on WinCE / Personal Java a couple of years back :-)

And there also exists a public (Ada 95) version of A# under sourceforge: http://sourceforge.net/projects/asharp

From: Rob Veenker <veenker@xs4all.nl>
Subject: Re: Ada / MS WinCE
Date: Tue, 06 May 2008 20:45:58 +0200
Newsgroups: comp.lang.ada

Just forgot to mention that GNATPro for .Net and A# also run on the .Net compact framework for WinCE! I even have Ada running on my cell phone :-)

From: Rob Veenker <veenker@xs4all.nl>
Subject: Re: Ada / MS WinCE
Date: Wed, 07 May 2008 20:23:03 +0200
Newsgroups: comp.lang.ada

> Would probably be nice to write a clear tutorial on how to do a Hello World with Ada with the compact framework on a cell phone!

There already exists a tutorial on A# for PDA's, last held at SigAda 2006. The only change for cell phones is that cell phones have a limited UI. You could however still use the M$ Visual studio (C#) to build the GUI so you know that part is OK and add Ada code in a .Net assembly. For deployment all you need to do is copy the A# runtime libraries onto your cell phone and add your own application assembly.

Not too much magic here :-)

To keep it all in Ada, you can probably also use Rapid for the GUI part but I am not sure all the calls performed by rapid are supported in the compact framework.

[See also "Programming a PDA with Ada" in AUJ 25-4 (Dec 2006), p.186. — su]

From: Philippe Bertin <philippe.bertin@telenet.be>
Subject: Re: Ada / MS WinCE
Date: Fri, 2 May 2008 03:03:13 −0700 (PDT)
Newsgroups: comp.lang.ada

> On my hardware there is an ARM CPU and I have a C/C++ Toolchain working together with a Linux BSP. What would be the steps to compile the Ada runtime and tailor the toolchain so it compiles Ada applications ?

In WinCE there's always a platform builder involved. Depending on your target platform, the separate (C/C++) header files may or may not be available. This depends on how the target platform was built. This "target platform build" step will most probably be done by some senior programmer or some software project responsible, who knows which libraries/drivers are available and which ones aren't on the target platform.

Now C/C++ header file availability isn't of that much interest to an Ada programmer. But with the above in mind, I doubt if there can be any "globally usable" WinCE port available for Ada, as every single port should be able to provide you the platform library (sub)set which is appropriate for the specific target platform...

References to Publications

Circuit Cellar — "Robotics with Ada 95"

From: Bill
Subject: Article: Ada For Robotics
Date: Fri, 21 Mar 2008 06:48:06 −0600
Newsgroups: comp.lang.ada

I'm a one-time embedded systems developer who has been stuck in the bug-infested world of IT for some time. I have been wanting to get back into embedded systems but haven't seen anything interesting enough to cause me to jump back in ... until I read the March

2008 issue of Circuit Cellar magazine. (www.circuitcellar.com) If you want to look into Ada, this article is a great place to start.

From: Richard Riehle <rdriehle@nps.edu>
Subject: Re: Article: Ada For Robotics
Date: Thu, 27 Mar 2008 07:43:05 −0700
Newsgroups: comp.lang.ada

This application description is one of the best testimonials to the benefits of Ada that I have seen in years.

Ada software and compiler publishers should request reprints and make them available at conferences such as Embedded Systems Conference, and the annual STC conference in Salt Lake City.

I am forwarding this to the professor at our school (NPS) who has taken charge of our robotics program. He is planning to use Erlang, but maybe this will encourage him to take another look at Ada.

Slashdot — Discussion about Ada

From: Jerry <lanceboyle@qwest.net>
Subject: Ada discussion at slashdot
Date: Tue, 15 Apr 2008 15:49:11 −0700 (PDT)
Newsgroups: comp.lang.ada

There is a discussion of Ada going on at slashdot:

http://developers.slashdot.org/article.pl?sid=08/04/15/1554234

It seems to have been inspired by the article which was mentioned here yesterday [See "GCN — The Return of Ada" in this issue —su]

The majority of the discussion is very positive for Ada.

Military Embedded Systems — "Ada Matters!"

From: AdaCore Press Center
Subject: Ada Matters!
Date: Tuesday May 13, 2008
RSS: http://www.adacore.com/2008/05/13/ada-matters/

Military & Aerospace Electronics — "Software code and COTS"

From: AdaCore Press Center
Date: Tuesday May 13, 2008
Subject: Software code and COTS
RSS: http://www.adacore.com/2008/05/13/software-code-and-cots/

Embedded Computing Design — "Don't blow your stack"

Date: Thursday May 15, 2008
Subject: Don't blow your stack
RSS: http://www.adacore.com/2008/05/15/static-stack-analysis-for-high-integrity-systems/

GCN — "The Return of Ada"

From: AdaCore Developer Center
Date: Monday April 14, 2008
Subject: The Return of Ada
RSS: http://www.adacore.com/2008/04/14/the-return-of-ada/

A very nice article entitled the "Return of Ada" has been published in Government Computer News. In it, the author highlights some of the recent contract wins and successes that made the choice to use the Ada programming language. To view the article, please [...] visit: http://www.gcn.com/print/278/46116-1.html

Computerworld — "The A-Z of Programming Languages: Ada"

From: OSNews
Subject: The A-Z of Programming Languages: Ada
Date: June 4, 2008
RSS: http://www.osnews.com/story/19824

Computerworld is undertaking a series of investigations into the most widely-used programming languages. Previously they have spoken to Alfred v. Aho of AWK fame, and Chet Ramey about his experience maintaining Bash. In this article, they chat with S. Tucker Taft, Chairman and CTO of SofCheck. Taft has been heavily involved in the Ada 1995 and 2005 revisions, and still works with the language today as both a designer and user. Computerworld spoke to Taft to learn more about the development and maintenance of Ada.

http://www.techworld.com.au/article/223388/-z_programming_languages_ada?pp=1

Doctor Dobb's Journal — "Tunny, Colossus and Ada: Keeping an Open Mind"

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 16 May 2008 03:56:45 −0700 (PDT)
Subject: Ada featured in Doctor Dobb's Journal
Newsgroups: comp.lang.ada

An opinion piece from Joachim Schüeth, the winner of the British National Museum of Computing's Colossus Cipher Challenge.

http://www.ddj.com/architect/207800151

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Newsgroups: comp.lang.ada
Subject: Re: Ada featured in Doctor Dobb's Journal
Date: Fri, 16 May 2008 06:04:59 −0700 (PDT)

> Good stuff. Another statement of what I have read here so often, that Ada is very good at letting one model the problem space rather than having to work backwards from the solution space.

Aye. The article has been slashdotted, too, and there are surprisingly few bad comments so far. And I'm apparently not the only one believing in an upsurge in the use of Ada. Way to go.

US National Academies

From: Praxis High Integrity Systems News
Subject: US agencies say software best practice is to be found in the UK
Date: August 10, 2007
URL: http://www.praxis-his.com/news/usSoftware.asp

Praxis has received further recognition and praise for its work in advanced software engineering, following two recent reports on software dependability and assurance from the United States (US).

The first, "Software for Dependable Systems: Sufficient Evidence?"(link 1 below) from the US National Academies, calls for an evidence-based approach to software assurance — something that Praxis has long practised and advocated in the UK. The report refers to several Praxis projects as examples of best practice in software engineering, particularly in the areas of formal methods and programming language design and verification.

The second report, "Software Security Assurance"(link 2 below) from the US Defense Technical Information Center, also identifies Praxis for its work in developing the Correctness by Construction software process, the SPARK programming language, and the SafSec assurance methodology, which Praxis developed for the UK MoD.

Keith Williams, Managing Director at Praxis, commented, "It's very encouraging to see these significant US bodies identify the technologies and practices that we have advocated for many years. In particular, the call for a more evidence- and product-based approach to software assurance is very welcome."

References

1. [http://www7.nationalacademies.org/CSTB/project_dependable.html]

2. [http://iac.dtic.mil/iatac/press_SOAR.html]

AdaCore Spring newsletter

From: AdaCore Developer Center
Date: Tuesday May 13, 2008
Subject: Spring newsletter available
RSS: http://www.adacore.com/2008/05/13/spring-newsletter-available-2/

The latest edition of the GNAT Pro Insider newsletter has been published and is available for download at:

www.adacore.com/category/press-center/newsletters

This issue features:

- New Release of GNAT Programming Studio
- Contract Award for Coverage Analysis Project
- Current Releases
- In the Pipeline
- Academia Corner
- Interview with Emmanuel Briot
- Webinar Schedule
- Technology Corner:Pragmas Precondition and Postcondition
- Conferences/Events

SPARK Brochure

From: SPARKAda.com
Subject: New SPARK Brochure — March 2008
Date: March 2008
URL: http://www.praxis-his.com/sparkada/

New SPARK Brochure — March 2008

Download the new SPARK Language and Toolset brochure

http://www.praxis-his.com/sparkada/pdfs/SPARK_Brochure.pdf

(8 pages, PDF, 470kb)

Presentation of Project Coverage

From: AdaCore Developer Center
Subject: Coverage Project
Date: Wednesday June 4, 2008
RSS: http://www.adacore.com/2008/06/04/coverage-project/

Thanks to French public funds, the next generation of Free Software code coverage tools is on its way. "Project Coverage" will produce a Free Software coverage analysis toolset together with the ability to generate artifacts that allow the tools to be used for safety-critical software projects undergoing a DO-178B software audit process for all levels of criticality.

While an important target use of the coverage toolset is safety-critical embedded applications, the design of the tools allows its use in non safety-critical projects.

Beyond the production of useful tools and certification material for industrial users, an important goal is to raise awareness and interest about safety-critical and certification issues in the Free Software/Open Source community.

The key insight of "Project Coverage" is as follows: code coverage can greatly benefit from recent advances in hardware virtualization technology as promoted, for instance, by QEMU. The attached slides give a presentation of the technical scope of the project.

http://www.adacore.com/wp-content/uploads/2008/06/project_coverage.pdf

[See also "AdaCore — Project Coverage" in this issue. —su]

From: AdaCore Developer Center
Subject: Coverage and Free Software
Date: Wednesday June 4, 2008
RSS: http://www.adacore.com/2008/06/04/coverage-and-free-software/

A recent paper by Franco Gasperoni describing how a Free Software toolset (Coverage) and virtualization technology (QEMU) can be used effectively to assure code coverage in the development of software applications. While an important target use of the coverage toolset is safety-critical embedded applications, the design of the tools allows its use in non safety-critical projects.

http://www.adacore.com/wp-content/uploads/2008/06/coverage_and_free_software.pdf

SSTC 2008 Tutorial

From: AdaCore Developer Center
Date: Tuesday April 29, 2008
Subject: Designing Safe and Secure Systems
RSS: http://www.adacore.com/2008/04/29/designing-safe-and-secure-systems/

Ben Brosgol's tutorial at SSTC 2008, entitled "Safety and Security: An Analysis of Certification Issues and Technologies for High-Integrity Software".

Today's interconnected critical systems must be both safe and secure; software developers and decision makers need to understand the operative certification standards and their implications on technology choice and system development. This presentation first summarizes the DO-178B avionics safety standard and the Common Criteria / Common Evaluation Methodology security standard. It identifies the requirements that these standards impose on programming language technology and development tools, and explains how safety and security considerations are similar and how they differ. It describes how modern programming language features such as Object-Oriented Programming affect safety and security certification, and assesses several current language family approaches — C / C++, Ada / SPARK, and Java — against safety and security requirements.

[See http://www.adacore.com/wp-content/uploads/2008/05/brosgolpresentation-sstc2008.pdf —su]

AdaCore — Multiple exhibitions

From: AdaCore Press Center
Date: Monday May 5, 2008
Subject: DASIA 2008
RSS: http://www.adacore.com/2008/05/05/dasia-2008/

AdaCore will be exhibiting at this event.

From: AdaCore Press Center
Date: Monday June 2, 2008
Subject: Eclipse Summit 2008
RSS: http://www.adacore.com/2008/06/02/eclipse-summit-2008/

AdaCore is sponsor of this event.

From: AdaCore Press Center
Date: Monday June 2, 2008
Subject: IET Systems Safety 2008
RSS: http://www.adacore.com/2008/06/02/iet-systems-safety-2008/

AdaCore is the major sponsor of this event.

From: AdaCore Press Center
Subject: SAFECOMP 2008
Date: Monday June 2, 2008
RSS: http://www.adacore.com/2008/06/02/safecomp-2008/

AdaCore is the main industrial sponsor of this event.

Robert Dewar, AdaCore CEO and President, will be presenting a talk at the Safety and Security event held on Thursday September 25 (details to follow).

From: AdaCore Press Center
Date: Monday March 17, 2008
Subject: GCC Developers' Summit
RSS: http://www.adacore.com/2008/03/17/gcc-developers-summit/

AdaCore is a major sponsor of this event and will be presenting an "Ada Compiler Tutorial".

From: AdaCore Press Center
Date: Wednesday March 12, 2008
Subject: User Experience of Tools for Safety-Critical Systems
RSS: http://www.adacore.com/2008/03/12/user-experience-of-tools-for-safety-critical-systems/

Organized by the Safety-Critical Systems Club, this event will focus on first-hand experience of applying tools to the development of safety-critical systems. AdaCore is the major sponsor of this event.

From: AdaCore Press Center
Date: Thursday May 8, 2008
Subject: Real-Time & Embedded Computing Conference (RTECC)
RSS: http://www.adacore.com/2008/05/08/real-time-embedded-computing-conference-rtecc/

AdaCore will be exhibiting at this event.

Ada Inside

Who's using Ada in industry?

From: Michael Feldman <mfeldman@seas.gwu.edu>
Subject: Who's using Ada in industry?
Date: Thu, 20 Mar 2008 23:08:04 −0000
Organization: The George Washington University
Newsgroups: comp.lang.ada

After several years of inactivity with the Ada project list, I've started to maintain it actively again. I've put a revised version online at

http://www.seas.gwu.edu/~mfeldman/ada-project-summary.html

I've changed the "look" a bit, added a few listings, changed the order of the project groupings, and removed all the dead links (without removing the associated listings). There aren't too many links now, but I think they are all valid.

Now I need your help. Please look at this long and interesting list, and let me know if you have anything to add to it (or delete if necessary). If you can provide a valid link for a project that doesn't have one, that would be great.

This list will best serve the community if its listings refer to actual applications of Ada that are either fielded or under active implementation. I don't want to pad the list with tools, compilers, libraries, etc., but rather to respond to the question "who is actually using Ada in industry?".

If you are close enough to a project to know there's "Ada inside", please let me hear from you. In the past, some listings have come from "anonymous sources"; you can be assured of my discretion; I always respect the confidentiality of my informants.

As always, please help make this list as complete and accurate as possible; it's good for all of us.

Thanks very much in advance for your help and interest!

Michael Feldman

Chairman, ACM SIGAda Education Working Group

Professor Emeritus, Department of Computer Science

The George Washington University

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Fri, 21 Mar 2008 21:28:41 +0200
Subject: Re: Who's using Ada in industry?
Newsgroups: comp.lang.ada

I suspect that this exclusion of the SW tool industry from the definition of "industry" may focus the list on the traditional Ada application areas — military and aerospace — and exclude small or start-up Ada adopters in other areas. This could reinforce prejudice against Ada in other areas and especially in the SW tools area.

I understand that a vendor of Ada compilers may be suspected of bias if the vendor chooses to write the compilers in Ada. But the same should not happen when a C compiler, or a language-neutral tool, is written in Ada. Moreover, I believe that the reputation of Java has been boosted by the fact that many tools for Java development are written in Java. Why should the same not be true for Ada? If SW tool projects were listed under their own heading — perhaps as the last group — they should not detract from the impact of the other projects on the list.

I admit that I am biased, as my own project (and that of at least one other company in the same area, that I know of) are SW tools and thus apparently not wanted on the list. Perhaps only large, expensive projects are wanted? In that case it would be clearer to say so.

From: Michael Feldman <mfeldman@seas.gwu.edu>
Organization: The George Washington University
Date: Thu, 01 May 2008 13:11:50 −0500
Subject: "Who's Using Ada" — May edition is online
Newsgroups: comp.lang.ada

I've just put up the May edition of the Ada project catalog. Thank you all very much for the numerous additions and corrections. The list of defense-related projects is significantly longer than before, but other categories are also growing as people send me tips. I'm glad to see growth in the desktop apps group; I suspect there are more of those I haven't learned of yet.

I've decided to leave the previous editions online for comparison purposes. The basic link above will always point to the most recent; I've identified the older ones by adding a year and month to the file name. For example, April 2008 is http://www.seas.gwu.edu/~mfeldman/ada-project-summary-0804.html

I'm still checking out some tips and possible links; they'll appear in the June edition.

Have a good May; see you again in June.

AdaCore — New Avionics Flight Control Software

From: AdaCore Press Center
Date: Wednesday March 5, 2008
Subject: AdaCore and Mistral Solutions Selected by Indian Government to Build New Avionics Flight Control Software
RSS: http://www.adacore.com/2008/03/05/mistral-solutions/

NEW YORK, AMSTERDAM, Netherlands, and BANGALORE, India, March 5, 2008 — Avionics 2008 — AdaCore, provider of the highest quality Ada tools and support services, today announced that the Indian Aeronautical Development Establishment (ADE) has chosen AdaCore's GNAT Pro High-Integrity Edition for DO-178B Ada environment to be used by AdaCore's partner Mistral Solutions to create new safety-critical flight control systems that will underpin advanced Indian defense programs.

The partnership was chosen over the competition due to the combination of Mistral's development experience and strong customer support, the safety-critical aspects of the Ada language, and AdaCore's knowledge, superior development environment and close integration with Wind River's operating software. This is the first ADE project to use the Ada language.

Mistral will use GNAT Pro High-Integrity Edition for DO-178B on the project, which aims to develop avionics flight control systems that will become the standard within multiple applications for the Indian Ministry of Defence. It began in April 2007 and is scheduled to last 14 months. The project will run on VME PPC-based single board computers using Wind River Systems' VxWorks 6.x operating system.

"This new project demonstrates that Ada is now the language of choice for developing safety-critical avionics systems across the globe," said Cyrille Comar, Managing Director, AdaCore Europe. "Working with our partner Mistral Solutions, AdaCore was able to deliver a superior development environment that is both easy to use and provides the high level of protection necessary for mission-critical applications such as flight control systems."

"This is yet another great example of a new avionics project leveraging the industry-proven combination of Wind River's VxWorks RTOS and AdaCore's GNAT Pro development environment," said Rob Hoffman, General Manager of Aerospace and Defense, Wind River. "Combining Wind River's VxWorks product platform and our worldwide support organization with AdaCore's software, we are ensuring that organizations like the Indian ADE benefit from the industry's most robust development platforms for high-reliability systems."

ADE is part of the Indian Defence Research and Development Organisation (DRDO), which undertakes design and development leading to the production of world-class flight control systems and equipment to meet the needs and requirements of the three Indian armed services. DRDO is working in various

areas of military technology, which include aeronautics, armaments, combat vehicles, electronics, instrumentation engineering systems, missiles, materials, naval systems, advanced computing, simulation and life sciences.

"This is the first major Ada project we have worked on, and when it came to developing such a safety-critical application, AdaCore was the natural choice to partner with," said Mujahid Alam, Vice President — Sales, Mistral Solutions. "From our extensive research, AdaCore stands out as the foremost provider of Ada technology, and we have been very impressed with AdaCore's knowledge, product strengths and strong integration with Wind River's technology. The success of this project is testament to the close working relationship we have built up, combined with our own development and customer support strengths."

GNAT Pro High-Integrity Edition for DO-178B supports the development of safety-critical and security-critical applications for embedded systems, servers and workstations. The GNAT Pro development environment combines market-leading technology with an expert support system to provide a natural solution where efficient and reliable code is critical.

About Mistral

Mistral is an ISO 9001:2000 certified and CMMi Level 3 appraised premier product realization Company providing end-to-end services for product design and development in the embedded space. Mistral offers expert design and development services covering hardware and software, customizable product designs and IP's, System Integration and COTS Solutions that improve quality and accelerate time-to-market for a broad range of embedded systems. Mistral has forged successful partnerships with leading providers of embedded solutions, which has enabled the company to provide its clients with the finest technology solutions based on the world's best platforms. For details, please visit http://www.mistralsolutions.com/

AdaCore — Project Coverage

From: AdaCore Press Center
Subject: AdaCore Announces "Project Coverage"
Date: Thursday June 5, 2008
RSS: http://www.adacore.com/2008/06/05/coverage/

The first Open Source code coverage project for DO-178B and safety-critical systems

Paris and New York, June 5, 2008 — AdaCore, together with Open Wide, ENST, and LIP6, and with financial support from French public funds, today announced the initiation of Project Coverage. The first Open Source project of its kind, Project Coverage will produce a Free Software coverage analysis toolset together with artifacts that allow the tools to be used by developers of safety-critical and mission-critical projects, including systems that need to be certified under safety standards such as DO-178B.

"Being strongly rooted in Free Software and having many customers in the Avionics and DO-178B domain, AdaCore was instrumental in getting this project off the ground," said Roberto Di Cosmo, President of the Free and Open Source Software group of System@tic, the R&D competitive cluster out of which Project Coverage has grown.

The key insight of Project Coverage is that code coverage can greatly benefit from recent advances in hardware virtualization and emulation technologies.

"By virtualizing the target hardware, Project Coverage tools can execute the target binary code unmodified on a host computer, such as a GNU Linux or Windows machine, and collect binary branch information," said Olivier Hainque, technical lead of Project Coverage at AdaCore. "The collected information is then analyzed off-line and mapped back to the original sources thanks to the debugging information contained in the executable."

"Our virtualization technology is based on QEMU. We are extending it, first to output execution traces, including binary branch coverage information, and second to make it usable in industrial contexts typically found in the avionics domain," continued Hainque.

"Because QEMU works by compiling the target object code into the host object code, virtualization is more effective than direct execution on the target," said Tristan Gingold, Senior Software Engineer at AdaCore and QEMU expert. "The speed advantage of the host over the target makes up for the loss in emulation performance and you gain the convenience and availability of the host environment over the target."

Beyond the production of useful tools and certification material for industrial users, an important goal of the project is to raise awareness and interest about safety-critical and certification issues in the Free Software/Open Source community.

"It's all about cross fertilization between the DO-178B and Free Software/Open Source communities," said Cyrille Comar, the AdaCore representative on the DO-178C committee. "The DO-178B community, with its approach anchored in requirements-based testing, has shown us that the source isn't everything in safety-critical systems, while the Open Source community has shown us that being open and having high-quality and widely available tools is essential to extend the benefit of state-of-the-art technologies from niche markets to a wider audience of software developers.

"The approach put forth by Project Coverage features several strong points," said Robert Dewar, President and CEO of AdaCore. "Project Coverage tools will be easy to use and deploy since they run on the host computer. They will be independent of the programming language and will work for Ada, C, and C++. Project Coverage tools are also designed to be non-intrusive and work directly with the final executable. No specialized hardware will be required to extract coverage information."

Project Coverage tools will be freely available, and industrial users will have the option to purchase high-quality professional support together with DO-178B qualification material.

In summary Project Coverage smartly combines several significant but independent trends in today's software technology landscape (Free Software/Open Source, Virtualization, DO-178B qualification, etc.) producing a unique code coverage solution that safety-critical and non safety-critical developers can use in their projects.

For more information on Project Coverage please go to www.adacore.com/home/company/development_projects

About Project Coverage

Project Coverage is partially funded by the regional authorities of Paris and the Ile-de-France district and the French Ministry of Industry, under the auspices of the Free & Open Source Software group headed by Roberto Di Cosmo in System@tic, the R&D competitive cluster located in Paris and its surroundings. The institutions participating in Project Coverage include AdaCore, Open Wide (who will provide an avionics test bed and study the impact of MIL-STD-1553, ARINC 629, and similar avionics standards for Project Coverage), ENST, and LIP6, who will generalize the Project Coverage approach to distributed systems and languages running on a virtual machine.

[See also "Presentation of Project Coverage" in this issue. —su]

Artisan Software — BAE Systems Ada Profile

From: Artisan Software Press Releases
Subject: Objektum Solutions and Artisan team with BAE Systems to resolve the HOOD-to-UML migration challenge on the Nimrod project
Date:Wednesday, March 05, 2008

URL: http://www.artisansw.com/news/press_release_details.aspx?pressReleaseID=185

Avionics Show 2008, Amsterdam— Wednesday, March 05, 2008 — Objektum Solutions' HOODbridge utility migrates HOOD design information to Artisan Studio UML models without any loss of design data or integrity

Avionics Show 2008, Amsterdam — 5th–6th March 2008. Artisan Software Tools and Objektum Solutions are successfully working with BAE Systems to resolve the HOOD-to-UML migration challenge for its Nimrod project legacy designs. By providing a solution combining Objektum Solutions' HOODbridge utility, the Artisan Studio UML/SysML tool suite and the BAE Systems Ada Profile, the resulting solution significantly reduces the time it takes to migrate legacy designs into Artisan Studio ready for use.

The HOOD (Hierarchic Object Oriented Design) design methodology was established by the European Space Agency over 25 years ago. Although never considered a mainstream methodology, it was adopted and successfully employed by a variety of major avionics, aerospace, defense and nuclear organizations to support engineering programs requiring a robust methodology to support complex, mission-critical systems development. Given its age and limited adoption, this has resulted in both HOOD skills and tool support becoming limited.

BAE Systems recognized the need to move towards a more mainstream environment for complex, mission-critical design and development by migrating its legacy designs to the industry standard UML environment to continue the ongoing development and maintenance requirements to better fully support the Nimrod in the years to come.

During 2007, BAE Systems reviewed its various migration options and engaged Objektum Solutions, a company with extensive expertise in the HOOD methodology, to investigate ways in which its HOOD data could be migrated into Artisan Studio, which was already widely deployed within BAE Systems. The outcome of this engagement has resulted in Objektum Solutions developing the HOODbridge, a standalone utility to migrate HOOD design data into an Artisan Studio UML model without any loss of design data integrity.

BAE Systems used Artisan Studio's powerful, extensible and highly flexible features to develop an Ada Profile — where profile properties are first class citizens in the design. Additionally, code generation templates have been developed to generate code from the models. Using the Objektum Solutions HOODbridge with the BAE Systems Ada Profile and code generation templates, it has proven the ability to migrate HOOD designs into Artisan Studio along with the generation of source code consistent with BAE Systems' original COTS HOOD tool. The HOODbridge builds a readable, navigable intermediate view of the BAE Systems´ Nimrod designs which can be thoroughly checked before the UML model is generated. Once the design has been migrated from HOOD to UML, BAE Systems can then exploit the richer UML notation in order to maintain and further develop the models.

Having proven the viability of the general concept, the Nimrod project team at BAE Systems is now standardizing on HOODbridge and Artisan Studio as its tools of choice for the migration of HOOD software designs to UML. The team is continuing to refine and validate the migration process in order to prove its accuracy and reliability before beginning the migration task in earnest.

"With the HOODbridge/Artisan Studio HOOD-to-UML migration solution, the Nimrod project team at BAE Systems has been able to satisfy all of the objectives of its migration rationale," said Peter Kibble, Commercial Director at Artisan Software Tools. "The HOODbridge/Artisan Studio HOOD-to-UML migration solution provides BAE Systems with the ability to transition the Nimrod project to a mainstream methodology with comparative ease and at minimal cost, thus ensuring they have the ability to continue to develop and maintain their legacy systems. The HOODbridge solution provides protection against ageing tools with diminishing support, mitigates against the declining availability of HOOD expertise resulting from what is fast becoming an obsolete methodology and ultimately future-proofs the project through better alignment with industry standards."

About Artisan Software Tools

Artisan Software Tools is the leading independent supplier of premium-quality, industrial-grade, collaborative modeling tools for complex, mission-critical systems and software. Artisan's standards-based tool suite, Artisan Studio, provides comprehensive support for the leading industry standards, including OMG SysML, UML and Architectural Frameworks. It delivers on the promise of an integrated collaborative development environment — allowing systems and software engineering teams to work as one — from concept through to delivery and maintenance. Artisan has delivered a stable, robust working environment to thousands of users across an extensive range of complex applications in demanding sectors including military, aerospace and defense, automotive and transportation, telecommunications and electronics, and medical. Founded in 1997 with extensive venture capital backing, Artisan is headquartered in the USA and UK with offices in Germany and Italy, supported by a global distributor network. For more information visit: www.artisansw.com.

About Objektum Solutions

Objektum Solutions provides real-time, mission-critical software development and consulting services to aerospace and defense organizations involved in software intensive projects. The company is committed to providing flexible and innovative solutions by leveraging its extensive technical capabilities in order to meet the needs of clients. With a proven track record in real-time embedded and object-oriented software for a range of safety-critical projects, Objektum Solutions has experience of the processes, tools and techniques used in delivering projects to the relevant safety standards. This is combined with a long-term partnership approach working hand-in-hand with its clients to meet the project's overall objectives.

About HOODbridge

The HOODbridge works by defining mappings between the elements of a HOOD design and the Artisan Studio UML model in terms of a profile and browser structure, which has been specified by BAE Systems. For example, the HOOD hierarchy is represented in the browser, classes are created for each HOOD Object and HOOD Diagrams are drawn as a class diagram. The HOODbridge parses the HOOD design and builds a readable, navigable view of the HOOD design, allowing engineers to review the analysis before generation. UML artifacts are then produced for each of the HOOD design elements, based on the mapping rules. As elements are created, automatic checking ensures the integrity and consistency of the original HOOD design. Any errors are communicated to the engineer via the screen and a comprehensive log file.

Indirect Information on Ada Usage

[Extracts from and translations of job-ads and other postings illustrating Ada usage around the world. —su]

Job Description

The Senior Software Engineer will be responsible for development of embedded command, control, and communications software for the digitized battlefield. The software is deployed in [...] U.S. Army Aviation platforms. [...] It performs data messaging, provides wireless communication protocols, and interfaces with military communication equipment and aviation mission systems. Providing

advanced applications, the software manages current battlefield situational awareness data on friendly and enemy forces for pilot display, and manages current mission orders and digital map overlays.

The Senior Software Engineer will analyze requirements, create object oriented software designs, implement software, and perform unit and integration testing. Software will be implemented using Ada 95 and executes on a POSIX compliant operating systems.

Education: BS Computer Science or other field

Ada in Context

A4 formatted ARM

From: Randy Brukardt <randy@rrsoftware.com>
Subject: Re: A4 formatted ARM?
Date: Tue, 6 May 2008 15:48:20 −0500
Newsgroups: comp.lang.ada

> Does anyone have a PDF of the ARM formatted for A4 paper?

I don't think any such thing exists (and surely not officially). The ARM is formatted for a non-standard paper size (7x9") which was adopted for some reason for Ada 95. The AARM is formatted for US 8 1/2x11 paper (but it's unofficial in any case).

The best way to get a printed copy is the buy the Springer edition.

If you have more time than money, you could make one yourself:

(1) Download and compile the formatting tool (its found on ada-auth.org).

(2) Download the input source for the ARM.

(3) Generate an RTF form of the ARM.

(4) Load that into Word 2003 (other word processors *may* work, but only Word 97 and Word 2003 are *known* to work [and Word 2000 is known *not* to work]. I'll be trying OpenOffice 2.4 the next time I need to work on the ARM).

(5) Set the page size to A4.

(6) Apply the half-dozen hand patches. Regenerate the table-of-contents.

(7) Save in some other format than .rtf (saving as .rtf crashes every version of Word ever tried).

(8) Output as PDF using your favorite PDF writer.

If you have neither time nor money, forget the paper and use the HTML format; it is better anyway (because it has links).

From: John McCabe <john@assen.demon.co.uk>
Subject: Re: A4 formatted ARM?
Date: Thu, 08 May 2008 13:17:07 +0100
Newsgroups: comp.lang.ada

[...] the fruits of my labours are now available at:

http://www.assen.demon.co.uk/AdaRMA4/

From: Pascal Obry <pascal@obry.net>
Date: Tue, 06 May 2008 22:47:57 +0200
Subject: Re: A4 formatted ARM?
Newsgroups: comp.lang.ada

Printer? The ARM is so huge that I won't waste my paper for that especially since it will be a big stack when printed. Not very easy to handler... I would order the ARM if the goal is to have it on paper. I enjoy the printed edition of the ARM, printed on very thin paper it is not that big.

Interfacing with Objective-C

From: Ivan Levashew <octagram@bluebottle.com>
Subject: Re: Interfacing with Objective-C or Python
Date: Mon, 26 May 2008 15:25:15 +0700
Newsgroups: comp.lang.ada

> Is there any way to interface Ada with Objective-C (or Python) without having a wrapper? Has anyone done this before?

Objective-C from Apple and from FSF have different way of dispatching calls.

Apple's one has objc_sendMsg varargs function. The best way is to have a C wrapper for every kind of varargs invocation since one can't be sure if passing some kind of argument is equivalent to passing it as a varargs argument. But if one is sure, one can completely eliminate ObjC wrapper. objc_sendMsg must be imported several times for every combination of arguments one is going to use it with. One will also need to map selectors' string representation to their integer representation with NSSelectorFromString. Frozen constant will do.

Then one just calls objc_sendMsg (objc-class-or-object, selector, arg1, arg2, ...). And also reference counting: Controlled's Adjust/Finalize must be mapped to NeXTSTEP's retain/release. I don't know much about exception handling.

> without having a wrapper?

ObjectiveC is like C++. There are at least 3 incompatible implementations: Apple, FSF, POC (Portable Object Compiler). One either make a wrapper or go into details and make implementation-dependent binding. Cocoa APIs are partially duplicated in GNUStep and Cocotron, and Apple is rumored to release Cocoa for Windows [1] currently being tested on Safari for Windows. So it's usually better to make portable bindings. [...]

[1] http://www.roughlydrafted.com/RD/RDM.Tech.Q2.07/A35C23B9-BD22-4478-BC30-4111CFC360B5.html

From: Ivan Levashew <octagram@bluebottle.com>
Subject: Re: Interfacing with Objective-C or Python
Date: Mon, 26 May 2008 16:18:15 +0700
Newsgroups: comp.lang.ada

> But looks like a C-wrapper is inevitable, isn't it?

No, it isn't.

You just need to ensure that varargs invocation is identical to usual invocation.

Pragma Atomic is not related with tasking

From: Maciej Sobczak <maciej@msobczak.com>
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Date: Wed, 12 Mar 2008 09:28:42 −0700 (PDT)
Newsgroups: comp.lang.ada

[...]

> Consider the multi-core execution of two concurrent threads. One will modify a variable, another will read. One will keep the variable in the register, another will reload it from the memory. Volatile will force flushing to the memory. Otherwise thread 2 will read erroneous data.

No. Your perception of what and where is the “memory” is completely different from that of CPU(s). The volatile keyword can, at best, force the compiler to use some “memory address” for access. The problem is that between “memory address” and “physical memory” there is a long chain of funny things like asynchronous memory writer in CPU and a few layers of cache with complicated prefetch functionality. The cache is transparent so can be ignored, but the memory writer module and prefetch work *asynchronously* and this means that when you “read” and “write” your variables, you really don't read and write what you think — or rather not *when* you think. This happens out of the compiler's control (it's hardware), so “volatile” cannot help you - you can as well write it in assembly and still have the same problem.

You need memory barrier to ensure visibility between CPUs and volatile does not provide it (unless you have some funny compiler). To actually get the barrier, you have to either apply it by hand or use dedicated system services that do it for you (mutexes, etc.). Now, the best part — once you do it, volatile give you nothing.

No, the best part is this: volatile not only gives you nothing, it actually *slows the

program down*, because it prevents the compiler from applying some obvious optimizations when the given object is used many times within the critical section.

You don't want to slow your program down, do you? :-)

And last but not least — have you tried to use volatile with some C++ classes, like string, vector, map, etc.? Try it. It will not even compile, but certainly it is possible to use these classes with many threads.

Short version: don't use volatile for multithreading. It's not the correct tool to do the job. The correct ones are membars and these are part of dedicated system services. Use them.

From: Maciej Sobczak <maciej@msobczak.com>
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Date: Thu, 13 Mar 2008 02:40:47 −0700 (PDT)
Newsgroups: comp.lang.ada

[...] The problem is that Thread2 can see something different and there the assertion can fail. This can happen for two reasons:

1. If you write something to memory, you only instruct the CPU that you *want* something to be stored. It will do it — but not necessarily immediately and it can actually happen *some time later*. The problem is that if you store two values (separate store instructions), the CPU does not know that they are related — and can stored them physically in different order. The compiler has no control over it! It is the hardware that can reorder the writes to memory.

2. If you read something from memory, you can actually read something that is already in the cache — the value can be there already and *earlier*.

Both these mechanisms can make Thread2 see different order of modifications than that perceived by Thread1. Again, the compiler has no control over it. It is all in hardware. [...]

From: Simon Wright <simon.j.wright@mac.com>
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Date: Fri, 14 Mar 2008 21:54:26 +0000
Newsgroups: comp.lang.ada

I think that 'volatile' is only a way to tell the compiler that this object must be read/written directly each time; don't copy it to a register and manipulate values there.

Likewise 'atomic' says to read/write the whole thing with one machine operation. Probably because that's what the IO hardware requires; e.g., you must write all 32 bits at once, not just the only byte you've changed..

The only time this makes sense is with memory-mapped IO where the system must be designed so that this is sensible.

If you try it with memory with varying levels of hardware cache and with multiple cores you are going to be disappointed, I think.

From: Maciej Sobczak <maciej@msobczak.com>
Date: Sat, 15 Mar 2008 06:29:56 −0700 (PDT)
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Newsgroups: comp.lang.ada

[...]

> What also seems to be alarming is proliferation of architecture specificity into the programming techniques.

No, there is no specificity. I have an impression that you still try to keep the volatile mess and then declare that the platform specificity breaks your code, but the truth is totally inverse — you have broken code, period. That's it — don't expect compiler vendors to "fix the world" and penalize those who do things correctly. If you write correct code (yes, forget once and for all the volatile keyword), there is absolutely no architecture specificity to worry about.

> What will happen when architecture change?

Nothing. Correct programs will still work and broken programs will still be broken.

> C started as a language for very simple microprocessor architecture language. Now architectures outgrew what the language was originally designed for.

No, you can still target modern architectures with this "outdated" language. Just do it right. Interestingly, this also applies to Ada [...]

> Practitioners are simply patching things up trying to make 12-year's old pants to fit on 16-years old boy.

Yes, you are unfortunately right here. Practitioners have to worry about truckloads of broken code.

Ada research and new synchronization protocols

From: ME <abcdefg@nonodock.net>
Subject: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Date: Fri, 7 Mar 2008 22:04:09 −0800
Newsgroups: comp.lang.ada

As many of may have already noticed, there has been a tremendous furor over the lack of multicore support in the common languages like C and C++. I have been reading these articles in EE Times and elsewhere discussing this disaster with all the teeth gnashing and handwringing acting as though Ada never existed. Robert Dewar, our hero, has written an absolutely excellent article with a clever intro.

http://www.eetimes.com/news/design/showArticle.jhtml?articleID=206900265

From: Maciej Sobczak <maciej@msobczak.com>
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Date: Sat, 8 Mar 2008 14:11:04 −0800 (PST)
Newsgroups: comp.lang.ada

[...] Take for example lock-free algorithms. There is no visible research on this related to Ada, unlike Java and C++ (check on comp.programming.threads). Ada will most likely miss the "multicore revolution", unless it will *really* focus on performance — the point is that all this multicore hoopla revolves around performance, *exclusively*.

From: Ivan Levashew <octagram@bluebottle.com>
Date: Tue, 29 Apr 2008 14:15:36 +0700
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Newsgroups: comp.lang.ada

FYI: http://www.gidenstam.org/Ada/Non-Blocking/

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Sun, 09 Mar 2008 03:17:44 GMT
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Newsgroups: comp.lang.ada

> Ada hasn't "missed" the "multicore revolution". Quite the opposite, Ada has had multitasking built-in since the mid 80's and it works just fine on multicore platforms. (I know, I've been there, done that.) Perhaps someday one of those C variants you seem to prefer will have the same kind of advanced features.

Ada has had tasking since 1980 (Ada 80, MIL-STD 1815). It was significantly revised for Ada 83.

From: Peter C. Chapin <pchapin@sover.net>
Date: Thu, 13 Mar 2008 06:49:21 −0400
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Newsgroups: comp.lang.ada

This paper:

http://www.aristeia.com/Papers/DDJ_Jul_Aug_2004_revised.pdf

describes some of these issues in a C++ context and is written by two authors who probably know what they are talking

about. Their conclusion is that it is pretty much impossible to write correct multi-threaded code in C++ without (non-standard) compiler assistance. I'm paraphrasing here. It is my understanding, however, that this matter will be addressed in C++ 0x.

From: Georg Bauhaus <rm.tsoh.plus-bug.bauhaus@maps.futureapps.de>
Date: Sun, 09 Mar 2008 10:39:06 +0100
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Newsgroups: comp.lang.ada

> [...] C++ code using threading, OpenMP or MPI are just a mess. Impossible to maintain because C++ hackers seems to prefer working hard 6 month for gaining 2% of performance instead of buying a new computer with more core or adding some node on a cluster. Sorry, but I've seen that, horrible mess just because hacking C++ code seems fun to many people.

Then again, the multicore things have atomic updates built in which offer some opportunities that only happen to be part of the Ada language. Now this language feature becomes visible as part of the top selling CPUs... There is research on employing these CPU mechanisms for Ada use, but, IIUC, it is not *visible* on many sites that are visible to those interested in how to use new multicore CPUs. And Ada RTS/Library inclusion is not done yet, or is it?

Multicore algorithms can continue the great academic tradition of efficient algorithms. Aren't lock-free ones really a natural starting point? They also have their uses. IIRC, ready-made Communicating Sequential Processes has a lower visibility in CS than the basic critical section model.

Ada's tasking implementations are not currently known to be the best choice when an algorithm is about how to efficiently use the multicore CPU with word sized memory. The tasking protocol as implemented for x86 < Today turns out to be too heavy weight. The cost is far beyond 10%.

> The cherry on top of the cake is that your application can be ported to a new architecture without much trouble.

Gidenstam has ported his Primitives (Ada packages hiding the CPU's atomic updates) to at least Intel, SPARC, and MIPS. Built on top of the Primitives, he has a lock-free bounded buffer in queue mode...

[See http://www.gidenstam.org/Ada/Non-Blocking/ —su]

From: Pascal Obry <pascal@obry.net>
Date: Sun, 09 Mar 2008 15:54:59 +0100
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Newsgroups: comp.lang.ada

> I've seen 80x (eighty times) penalty when comparing Ada's protected objects with basic usage of mutexes in C++. 80x is not something to be taken lightly.

I've never seen such penalty. Maybe in theory or in a very specific part of the code. But let's compare the *final* application speed. I have gone this path in a medium simulation, the Ada implementation was slower in some part, the C++/OpenMP implementation was slower on some other part. The final application was running at same speed in Ada and C++ (well in fact the Ada implementation was a bit less than 1% faster than the C++ one).

Also, one point about the C++/MPI version (we also worked on a distributed version even if I don't have the final data, but speed was almost comparable) compared to the Ada Annex-E version. My co-worker were amazed at how fast I was able to re-configure the distributed application. Where it took days/weeks to change the MPI implementation, it took me hours to change the GLADE configuration file. Also, the facility to exchange Ada Containers objects across partitions was pretty amazing. No tweak, no hack, clean code, just plain Ada.

I know a group of C++ hackers still trying to come up with a clean solution to exchange objects (class instance) across nodes... Impossible to stream properly objects in C++, or you have to code almost all by hand! All these aspects are far more important to me than pure speed. I understand that this tradeoff can be different on some other applications, but I won't buy that this is majority of cases!

From: Ole-Hjalmar Kristensen <ole-hjalmar.kristensen@sun.com>
Date: 26 Mar 2008 14:49:40 +0100
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Organization: Sun Microsystems
Newsgroups: comp.lang.ada

If the compiler is smart enough to optimize this case, an entryless protected object would be a good building block.

The AARM states that “Entryless protected objects are intended to be treated roughly like atomic objects — each operation is indivisible with respect to other operations (unless both are reads), but such operations cannot be used to synchronize access to other nonvolatile shared variables”

From: Ole-Hjalmar Kristensen <ole-hjalmar.kristensen@sun.com>
Date: 27 Mar 2008 10:31:50 +0100
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Organization: Sun Microsystems
Newsgroups: comp.lang.ada

> You need some signalling for a shared hash table, otherwise reading a freshly-added object from a different thread might not give you the data you want (strictly speaking, even just comparing the might cause issues).

If you mean that it may be difficult to optimize, I agree, but I cannot agree that you need *more* than an entryless protected object to implement a hash table, since it guarantees that each operation on the object is indivisible.

The simplest case (for the programmer) is of course to put both key and value inside the protected object, Then a reader will either see the key, value pair as it was before the update or as it is after the update. The problem is that although reads within a procedure may be optimistic, the compiler probably needs to insert at least a spin lock during the actual update of the object.

What I was thinking of was to recognize the special case where the entryless protected object contains only a single entity which can be updated atomically with a compare and swap. In that case, you could skip the spin lock in the update phase and use CAS directly. In this case the key and the value would be in separate protected objects, and the implementation of the hash table could follow the pattern of the hash table you mentioned.

On the other hand, I cannot see any reason why Annex C just couldn't say that intrinsic subprograms for compare-amd-swap and similar machine operations shall be provided *if* they are available on a platform. That would at least save me the work of writing the bindings myself.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 28 Mar 2008 01:34:26 −0500
Subject: Re: Robert Dewar's great article about the Strengths of Ada over other langauges in multiprocessing!
Newsgroups: comp.lang.ada

> On the other hand, I cannot see any reason why Annex C just couldn't say that intrinsic subprograms for compare-amd-swap and similar machine operations shall be provided *if* they are available on a platform. That would at least save me the work of writing the bindings myself.

It does, actually. See C.1(11–16). It's “only” Implementation Advice, but that is necessary in any case, because the Standard can't require something it can't define.

A more interesting question is whether implementations follow that advice (in any useful manner).

Conference Calendar

This is a list of European and large, worldwide events that may be of interest to the Ada community. Further information on items marked ♦ is available in the Forthcoming Events section of the Journal. Items in larger font denote events with specific Ada focus. Items marked with ☺ denote events with close relation to Ada.

The information in this section is extracted from the on-line *Conferences and events for the international Ada community* at: http://www.cs.kuleuven.be/~dirk/ada-belgium/events/list.html on the Ada-Belgium Web site. These pages contain full announcements, calls for papers, calls for participation, programs, URLs, etc. and are updated regularly.

2008

July 01 — 8th **International Workshop on Worst-Case Execution Time Analysis** (WCET'2008), Prague, Czech Republic. In conjunction with the 20th ECRTS conference. Topics include: any issue related to timing analysis, such as Tools for timing analysis, Design for timing predictability, Integration of WCET analysis into the development process, etc.

☺ July 01-05 — 7th **International Symposium on Parallel and Distributed Computing** (ISPDC'2008), Krakow, Poland. Topics include: Parallel Computing; New Parallel System Concepts and Architectures; Distributed Systems Methodology and Networking; Parallel Programming Paradigms and APIs; Tools and Environments for Parallel Program Analysis; Task Scheduling and Load Balancing; Performance Management in Parallel and Distributed Systems; Distributed Software Components; Real-time Distributed and Parallel Systems; Security in Parallel and Distributed Systems; Fault Tolerance in Parallel and Distributed Systems; Parallel Scientific Computing and Large Scale Simulations; Parallel and Distributed Applications; etc.

July 06-13 — 35th **International Colloquium on Automata, Languages and Programming** (ICALP'2008), Reykjavik, Iceland. Topics include: Principles of Programming Languages; Formal Methods and Model Checking; Models of Concurrent and Distributed Systems; Models of Reactive Systems; Program Analysis and Transformation; Specification, Refinement and Verification; Type Systems and Theory; Foundations of Secure Systems and Architectures; Specifications, Verifications and Secure Programming; etc.

☺ July 06 — 1st **Interaction and Concurrency Experience** (ICE'2008). Topics include: Synchronous and Asynchronous Interactions in Concurrent Distributed Systems; models, logic and types for interactions; synchronous/asynchronous mechanisms; expressiveness results; timed and hybrid interactions; verification, analysis and tools; programming primitives for interactions;

☺ July 07-10 — 2008 **International Conference on Software Engineering Theory and Practice** (SETP'2008), Orlando, FL, USA. Topics include: Case studies, Component-based software engineering, Critical software engineering, Distributed and parallel software architectures, Education aspects of software engineering, Embedded software engineering, Model Driven Architecture (MDA), Model-oriented software engineering, Object-oriented methodologies, Program understanding, Programming languages, Quality issues, Real-time software engineering, Real-time software systems, Reliability, Reverse engineering, Software design patterns, Software maintenance, Software reuse, Software safety and reliability, Software security, Software specification, Software tools, Verification and validation of software, etc.

☺ July 07-11 — 22nd **European Conference on Object Oriented Programming** (ECOOP'2008), Paphos, Cyprus. Topics include: analysis, design methods and design patterns; concurrent, real-time or parallel systems; distributed systems; language design and implementation; programming environments and tools; type systems, formal methods; compatibility, software evolution; components, modularity; etc.

☺ July 07 — 18th **Doctoral Symposium and PhD Students Workshop**. Topics include: Design methods and design patterns; Concurrent, real-time or parallel systems; Distributed systems; Language design and implementation; Programming environments and tools; Type systems, formal methods; Software evolution; Components, Modularity; etc.

☺ July 07 **International Workshop on Advanced Software Development Tools and Techniques** (WASDeTT'2008). Topics include: What features in object-oriented languages make them easier to build tools for/with?

☺ July 07 **International Workshop on Object-Oriented Software Development for the Embedded World** (OOSDEW'2008). Topics include: object-oriented language features for embedded devices; software techniques and tools for optimizing code for embedded devices; etc.

☺ July 07 3rd **Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems** (ICOOOLPS'2008). Topics include: implementation of fundamental OOL features: inheritance (object layout, late binding, subtype test, ...), genericity (parametric types), memory management; runtime systems: compilers, linkers, etc; optimizations: static and dynamic analyses, threads and synchronization, etc; resource constraints: real-time systems, embedded systems; relevant choices and tradeoffs: separate compilation vs. global compilation, dynamic checking vs. proof-carrying code, annotations vs. no annotations, etc.

☺ July 08 7th **Workshop on Parallel/High-Performance Object-Oriented Scientific Computing** (POOSC'2006). Topics include: tried or proposed programming language alternatives to C++; issues specific to handling or abstracting parallelism, including the handling or abstraction of heterogeneous architectures; existing, developing, or proposed software; grand visions (of relevance); etc.

July 07-14 20th **International Conference on Computer Aided Verification** (CAV'2008), Princeton, USA. Topics include: Algorithms and tools for verifying models and implementations, Program analysis and software verification, Modeling and specification formalisms, Applications and case studies, Verification in industrial practice, etc.

☺ July 07-08 **Workshop on Exploiting Concurrency Efficiently and Correctly** ((EC)^2). Topics include: advances in programming languages and tools for developing concurrent software; programming constructs for concurrency; formalization of concurrency libraries; verification tools; introducing concurrency in education; etc.

July 15-18 9th **International Conference on Mathematics of Program Construction** (MPC'2008), Marseille (Luminy), France. Topics of interest range from algorithmics to support for program onstruction in programming languages and systems, such as type ystems, program analysis and transformation, programming-language emantics, etc.

July 16-18 **Static Analysis Symposium** (SAS'2008), Valencia, Spain. Topics include: abstract interpretation, compiler optimizations, control flow analysis, data flow analysis, model checking, program specialization, security analysis, type based analysis, verification systems, etc.

☺ July 16-18 **Workshop on The Impact of New Architectures on Parallel Programming** (IMPAR'2008), Sao Paulo, Brazil. Topics include: Parallel Languages and Libraries; Tools for parallel programming: debuggers, libraries and performance analyzers; Compilers; Scheduling; Performance Evaluation; Parallel Applications; etc.

July 20-24 **International Symposium on Software Testing and Analysis** (ISSTA'2008), Seattle, Washington.

☺ July 20 **International Workshop on Defects in Large Software Systems** (DEFECTS'2008). Topics include: Techniques to detect, locate, or predict defects; Empirical studies of defects; Types of defects that occur in software; Evolution of defects over time; Tools for post-deployment defect detection and reporting; Experience using certain techniques to identify or predict defects; etc.

☺ August 04-06 **International Workshop on Concurrent Programming Environment** (CoPE'2008), Hsinchu, Taiwan. Topics include: the foundations, tools and techniques, and experiences in practice for the development of concurrent software for embedded, real-time, multi-threaded, multi-core, multiprocessor, cluster, distributed, mobile, ubiquitous, or grid systems.

August 08-09 **International Symposium on Software Variability: a Programmers Perspective** (SVPP'2008), Brussels, Belgium. Topics include: Programming language abstractions for software variability; Modularization approaches for software variability; Guidelines to include software variability in

programs; Runtime support for software variability; Generative programming for software variability; etc.

August 18-21 27th **Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing** (PODC'2008), Toronto, Canada.

☺ August 25-27 14th IEEE **International Conference on Embedded and Real-Time Computing Systems and Applications** (RTCSA'2008), Kaohsiung, Taiwan. Topics include: Multi-thread programming for multi-core embedded platform, Embedded system design practices, Real-time scheduling, Timing analysis, Programming languages and run-time systems, Middleware systems, Design and analysis tools, Case studies and applications, etc.

☺ August 26-29 14th **European Conference on Parallel and Distributed Computing** (Euro-Par'2008), Las Palmas de Gran Canaria, Spain. Topics include: all aspects of parallel and distributed computing, such as Support tools and environments, High performance architectures and compilers, Parallel and distributed programming, Theory and algorithms for parallel computation, etc.

☺ August 26 EuroPar2008 - 2nd Workshop on Highly Parallel Processing on a Chip (HPPC'2008). Topics include: programming models; languages and software libraries; implementation techniques (e.g. multi-threading, work-stealing); support and performance tools, performance evaluation; parallel algorithms and applications; etc.; for/on highly parallel multi-core systems.

☺ August 27-29 12th **Brazilian Symposium on Programming Languages** (SBLP'2008), Fortaleza, Ceara, Brazil. Topics include: Programming language design and implementation; Design and implementation of programming language environments; Object-oriented programming languages; New programming models; Program transformations; Program analysis and verification; Compilation and interpretation techniques; etc.

September 01-03 5th **International Colloquium on Theoretical Aspects of Computing** (ICTAC'2008), Istanbul, Turkey. Topics include: software specification, refinement, verification; integration of theories, formal and engineering methods and tools; models of concurrency; parallel and distributed computing; real-time and embedded systems; principles and semantics of languages; case studies, theories, tools and experiments of verified systems; etc.

☺ September 03-05 7th **International Conference on Distributed and Parallel Systems** (DAPSYS'2008), Debrecen, Hungary. Topics include: Distributed and Grid middleware, Parallel and distributed programming languages and algorithms, Formal models for parallel and distributed computing, Software engineering and development tools, etc.

☺ September 07-10 9th **Conference on Communicating Process Architectures** (CPA'2008), York, UK. Topics include: Theoretical approaches to concurrency, and formal languages supporting these approaches, including the integration of existing formal notations; Modelling of, and model-driven development of concurrent software architectures; Verification and analysis of concurrent systems; Model-checking techniques and tools for development and analysis; Tools and languages for hardware-software co-design; Programming languages and environments for concurrent systems; Programming and implementation issues for concurrent languages, such as deadlock-freedom by design, starvation, and efficient inter-process communication architectures; System issues for programming languages supporting concurrency, such as multithreading kernels and interrupt architectures; Applications that exploit, or rely on, concurrency; etc.

☺ September 08-12 **International Conference on Parallel Processing** (ICPP'2008), Portland, Oregon, USA. Topics include: Compilers and Languages, Software Systems and Tools, etc.

☺ September 15-16 13th **International ERCIM Workshop on Formal Methods for Industrial Critical Systems** (FMICS'2008), L'Aquila, Italy. Topics include: Design, specification, code generation and testing based on formal methods; Verification and validation of complex, distributed, real-time systems and embedded systems; Verification and validation methods that address shortcomings of existing methods with respect to their industrial applicability; Tools for the development of formal design descriptions; Case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions; Application of formal methods in standardization and industrial forums; etc.

Sep 28 – Oct 03 11th **International Conference on Model-Driven Engineering Languages and Systems** (MoDELS'2008), Toulouse, France. Topics include: Model-driven engineering methodologies, approaches, languages and tools; Domain-specific modeling languages; Programming language and metaprogramming support for linking models to code; Models in the context of software evolution; Modeling languages and tools; Semantics of modeling languages; Modeling and analysis of real-time, embedded, and distributed systems; etc.

☺ Sep 29 MoDELS2008 - 1st **International Workshop on Model Based Architecting and Construction of Embedded Systems** (ACES-MB'2008). Topics include: model-oriented counterparts of specific design and implementation languages with particularly well-behaved semantics, such as synchronous languages and models (Lustre/SCADE, Signal/Polychrony, Esterel), time triggered models (TTA, Giotto), scheduling-oriented models (HRT-UML, Ada Ravenscar), etc. Deadline for submissions: July 15, 2008.

☺ October 06-08 27th IEEE **International Symposium on Reliable Distributed Systems** (SRDS'2008), Napoli, Italy. Topics include: High-confidence systems, Critical infrastructures, Distributed embedded systems, Formal methods and foundations for dependable distributed computing, etc.

October 06-10 2nd IFIP **Working Conference on Verified Software: Theories, Tools, Experiments** (VSTTE'2008), Toronto, Canada. Topics include: all aspects of verified software, theoretical as well as experimental, such as specification languages and case-studies, programming languages, language semantics, software design methods, automatic code generation, type systems, verification tools (static analysis, dynamic analysis, model checking, theorem proving, satisfiability), integrated verification environments, etc.

October 09-10 13th **Nordic Workshop on Secure IT Systems** (NordSec'2008), Copenhagen, Denmark. Topics include: Language-based Techniques for Security; New Ideas and Paradigms in Security; Security Education and Training; Software Security, Attacks, and Defenses; Trust and Trust Management; etc. Deadline for submissions: July 23, 2008.

☺ October 15 2008 **SPARK User Group meeting**, Bath, UK. Topics include: Formal Methods and DO-178C; The iFACTS project; Using SMT Solvers to Prove SPARK VCs; SPARK Update and Release 7.6 Highlights.

October 15-17 7th **International Conference on Software Methodologies, Tools, and Techniques** (SoMeT'2008), Sharjah, UAE. Topics include: Software methodologies, and tools for robust, reliable, non- fragile software design; Automatic software generation versus reuse, and legacy systems, source code analysis and manipulation; Intelligent software systems design, and software evolution techniques; Software optimization and formal methods for software design; Software security tools and techniques, and related Software Engineering models; End-user programming environment; etc.

October 15-18 15th **Working Conference on Reverse Engineering** (WCRE'2008), Antwerp, Belgium. Topics include: Program comprehension; Mining software repositories; Empirical studies in reverse engineering; Redocumenting legacy systems; Reverse engineering tool support; Reengineering to distributed architectures; Software architecture recovery; Program analysis and slicing; Program transformation and refactoring; etc.

☺ October 16-17 16th **International Conference on Real-Time and Network Systems** (RTNS'2008), Rennes, France. Topics include: Real-time system design and analysis (task and message scheduling, verification, formal methods, model-driven development, worst-case execution time estimation, distributed systems, fault-tolerance, security, ...); Software technologies for real-time systems (compilers, programming languages, middleware and component-based technologies, ...); Applications (automotive, avionics, telecommunications, process control, multimedia, inhouse entertainment, robotics); etc.

October 16-17 2nd **Junior Researcher Workshop on Real-Time Computing** (JRWRTC'2008). Topics include: Real-Time Distributed Systems, Middleware, Embedded Operating Systems, Real-Time Programming Language, Real-Time Software Engineering, System Development Tools, Worst-Case Execution Time Task Scheduling, etc. Deadline for submissions: September 1, 2008.

☺ October 19-23 23rd **Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2008), Nashville, USA. Topics include: new and better programming and design paradigms as well as practices. Deadline for submissions: July 2, 2008 (Development Program

Proposals, Student Research Competition, Onward! short papers and films, Doctoral Symposium and Student Volunteers).

☺ October 19 2nd **Workshop on Assessment of Contemporary Modularization Techniques** (ACoM.08). Topics include: Lessons learned from assessing new modularization techniques, Empirical studies and industrial experiences, Comparative studies between new modularization techniques and conventional ones, etc. Deadline for submissions: August 15, 2008.

☺ October 19-24 **Embedded Systems Week 2008** (ESWEEK'2008), Atlanta, Georgia, USA. Includes CASES'2008 (International Conference on Compilers, Architecture, and Synthesis for Embedded Systems), CODES+ISSS'2008 (International Conference on Hardware/Software Codesign and System Synthesis), EMSOFT'2008 (International Conference on Embedded Software).

♦ Oct 26-30 2008 ACM **SIGAda Annual International Conference** (SIGAda'2008), Portland, Oregon, USA. Sponsored by ACM SIGAda, in cooperation with SIGAPP, SIGCAS, SIGCSE, SIGPLAN, SIGSOFT, Ada-Europe, and Ada Resource Association (Cooperation approvals pending). Topics include: Transitioning to Ada 2005; Educational challenges for developing reliable, safe, secure software; Ada and SPARK in the classroom and student laboratory; Language selection for a high reliability system; Use of high reliability subsets or profiles such as MISRA C, Ravenscar, SPARK; High reliability standards and their issues; Software process and quality metrics; Analysis, testing, and validation; Use of ASIS for new Ada tool development; Mixed-language development; High-reliability development experience reports; Static analysis of code; Integrating COTS software components; System Architecture & Design; Information Assurance; Ada products certified against Common Criteria / Common Evaluation Methodology; etc.

☺ October 20-22 IMCSIT2008 - **International Workshop on Real-Time Software** (RTS'2008), Wisla, Poland. Topics include: Real-time system development, Scheduling, Safety, Reliability, Dependability, Standards and certification, Control software, Robotics and UAV, Software development tools, Model-based development, Real-time systems education, Related engineering curricula, etc.

November 09-15 16th ACM **SIGSOFT International Symposium on the Foundations of Software Engineering** (FSE-16), Atlanta, Georgia, USA. Topics include: Components and Middleware, Dependability (safety, security, reliability), Empirical Studies, Generative Programming, Software Reuse, Quality and Performance, Reengineering and Reverse Engineering, Specification and Verification, Tools and Environments. Deadline for submissions: August 11, 2008 (Doctoral Symposium abstracts).

☺ Nov 14 4th **International Workshop on Exception Handling** (WEH'2008). Topics include: Empirical studies of exception handling engineering; Design patterns and anti-patterns, architectural styles, and good programming practice cookbooks; Static analysis and testing of exception handling; Refactoring and evolution of exception handling code; Exceptions and variability management; Comparative studies of innovative exception handling techniques and conventional ones; etc. Deadline for paper submissions: August 10, 2008.

☺ Nov 10-12 10th **International Symposium on Distributed Objects, Middleware and Applications** (DOA'2008), Monterrey, Mexico. Topics include: Application case studies of distribution technologies; Development methodologies for distributed applications; Interoperability with other technologies; Reliability, fault tolerance, quality-of-service, and real time support; Scalability and adaptivity of distributed architectures; Software engineering for distributed middleware systems; etc.

☺ Nov 10-14 6th IEEE **International Conference on Software Engineering and Formal Methods** (SEFM'2008), Cape Town, South Africa. The aim is to advance the state of the art in formal methods, to scale up their application in software industry and to encourage their integration with practical engineering methods. Topics include: software specification, verification and validation; programming languages and type theory; program analysis; embedded systems; real-time and hybrid systems theory; software architectures and their description languages; light-weight formal methods; CASE tools and tool integration; applications of formal methods and industrial case studies; etc.

November 10-14 19th **International Symposium on Software Reliability Engineering** (ISSRE'2008), Seattle/Redmond, Washington, USA. Topics include: Reliability, availability, and safety of software systems; Validation and verification, testing; Software quality; Software security; Fault tolerance, survivability, and resilience of software systems; Open source software reliability engineering; Supporting tools and automation; Industry best practices; etc.; Empirical studies of any of the above topics. Deadline for submissions: July 1, 2008 (industry track abstracts), September 1, 2008 (industry track presentations), August 1, 2008 (government track, student track), August 10, 2008 (fast abstract).

☺ Nov 19-20 **Automotive - Safety & Security 2008**, Stuttgart, Germany. Organized by Gesellschaft für Informatik mit den Fachgruppen Ada, etc, and Ada-Deutschland. Topics include (in German): Zuverlässigkeit und Sicherheit für fahrbetriebskritische Software und IT-Systeme; Evaluation und Zertifizierung von Sicherheitseigenschaften automobiler Firmware/Software; Zuverlässige Echtzeit-Betriebssysteme; Fortschritte bei Normen und Standardisierungen; Zuverlässigkeit von Multi-Core-Architekturen; etc.

☺ Dec 01-04 9th **International Conference on Parallel and Distributed Computing, Applications, and Techniques** (PDCAT'2008), Dunedin, New Zealand. Topics include: Parallel/distributed architectures; Multi-core related technologies; Reliability, and fault-tolerance; Formal methods and programming languages; Software tools and environments; Parallelizing compilers; Component-based and OO Technology; Parallel/distributed algorithms; Task mapping and job scheduling; Security and privacy; etc.

December 01-05 ACM/IFIP/USENIX **9th International Middleware Conference** (Middleware'2008), Leuven, Belgium. Topics include: design, implementation, deployment, and evaluation of distributed system platforms and architectures for future computing and communication environments. Deadline for submissions: July 1, 2008 (industrial track papers), July 25, 2008 (doctoral symposium abstracts), August 1, 2008 (doctoral symposium papers).

☺ Dec 08-10 14th IEEE **International Conference on Parallel and Distributed Systems** (ICPADS'2008), Melbourne, Australia. Topics include: Parallel and Distributed Applications and Algorithms; Multi-core and Multithreaded Architectures; Resource Management and Scheduling; Dependable and Trustworthy Computing and Systems; Real-Time Systems; etc.

December 03-05 11th IEEE **International Symposium on High Assurance Systems Engineering** (HASE'2008), Nanjing, China. Topics include: Design and development of highly reliable, survivable, secure, safe, and time-assured systems; Policies for reliability, safety, security, integrity, privacy, and confidentiality of high assurance systems; Formal specification, specification validation, testing, and model checking for high assurance systems; High assurance software architecture and design; etc. Deadline for submissions: July 10, 2008 (fast abstracts).

December 10 Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

☺ Dec 10-12 6th **International Symposium on Parallel and Distributed Processing with Applications** (ISPA'2008), Sydney, Australia. Topics include: all aspects of Parallel/Distributed Computing and Networking, and their applications, such as Parallel/distributed system architectures, Tools and environments for software development, Distributed systems and applications, Reliability, fault-tolerance, and security, etc.

2009

☺ March 04-07 40th ACM **Technical Symposium on Computer Science Education** (SIGCSE'2009), Chattanooga, Tennessee, USA.

☺ March 08-12 **SAC2009 - Track on Software Engineering** (SE'2009), Honolulu, Hawaii, USA. Topics include: Component-Based Development and Reuse; Safety and Security Dependability and Reliability; Fault Tolerance and Availability; Design Patterns; Standards; Maintenance and Reverse Engineering; Verification, Validation, and Analysis; Formal Methods and Theories; Empirical Studies and Industrial Best Practices; Applications and Tools; Distributed, Embedded, Real-Time, High Performance, and Highly Dependable Systems; etc. Deadline for paper submissions: August 16, 2008

March 22-29 12th **European Joint Conferences on Theory and Practice of Software** (ETAPS'2009), York, UK. Deadline for submissions: October 2, 2008 (abstracts), October 9, 2008 (papers).

March 24 **Ada Conference UK 2009**, London, UK. This event is organised to promote awareness of the Ada programming language, and to highlight the increased relevance of Ada in safety- and security-critical programming. Since its inception, Ada has been successful in systems where reliability is essential. Its application domains include aeronautics, air traffic control, aerospace, simulation, shipboard systems, railway systems, communications, banking and many others.

☺ May 16-24 31st **International Conference on Software Engineering** (ICSE'2009), Vancouver, Canada. Topics include: Specification and Verification; Software Architecture and Design; Patterns and Frameworks; Reverse Engineering, Refactoring, and Evolution; Tools and Environments; Empirical Software Engineering; Development Paradigms and Software Processes; Component-based Software Engineering; Model Driven Engineering; Distributed Systems and Middleware; Embedded System; Open Standards and Certification; Software Economics; Dependability (safety, security, reliability); Case Studies and Experience Reports; etc. Deadline for submissions: August 29, 2008 (research abstracts), September 5, 2008 (research papers), September 15, 2008 (workshops, tutorials), October 10, 2008 (Software Engineering in Practice), November 2008 (SCORE), November 24, 2008 (research demonstrations), December 5, 2008 (Emerging Results track), December 12, 2008 (doctoral symposium).

♦ June 08-12: 14th **International Conference on Reliable Software Technologies – Ada-Europe'2009**, Brest, France.

December 10 Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

Association for Computing Machinery

SIGAda 2008

Annual International Conference on the Ada Programming Language

October 26-30, 2008, Portland, Oregon, USA

University Place Hotel and Conference Center
310 SW Lincoln St., Portland, Oregon 97201 (USA)

Portland and Mt. Hood; Streetcar Near Conference Hotel

Advance Program coming in July 2008. Visit
http://www.sigada.org/conf/sigada2008
for details.

SIGAda 2008
Annual International Conference on the Ada Programming Language
October 26-30, 2008, Portland, Oregon, USA
Special Anniversary Keynote Addresses

From Strawman to Ada 2005: a Socio-Technical Retrospective

Ben Brosgol, *Senior Technical Staff, Adacore*

Dr. Benjamin Brosgol, a senior member of the technical staff of AdaCore, has been involved with programming language design and implementation for more than 25 years, concentrating on languages and technologies for high-reliability systems. He led the development of the "Red" language candidate at Intermetrics, participated in the design of both Ada 83 and Ada 95, and was editor of the Safety and Security Annex of the Ada 95 standard.

Under Sun Microsystems' Java Community Process Dr. Brosgol was a member of the Expert Group for JSR-001 (Real-Time Specification for Java, or "RTSJ"), and he is currently a member of the Expert Groups for JSR-282 (RTSJ v1.1) and JSR-302 (Safety-Critical Java Technology). Dr. Brosgol is a past chair of the ACM Special Interest Group on Ada (SIGAda). He has spoken widely on safety-critical software technology. He holds a B.A. in Mathematics from Amherst College, and M.S. and Ph.D. degrees in Applied Mathematics from Harvard University.

30 Years after Steelman: Does DoD Still Have a Software Crisis?

Joyce Tokar, *President, Pyrrhus Software*

Joyce Tokar is the President of Pyrrhus Software, a software consultancy and training company. Over the past 20 years, Dr. Tokar has been working in the area of mission and safety critical, real-time, and embedded software systems. She has been involved in research and development in the areas of software and systems architectures, high level computing languages such as Ada, Ada 95, C/C++, and Java, and real-time analysis methodologies. During this time she has co-authored the Society of Automotive Engineering (SAE) Architecture Analysis and Design Language (AADL) standard. She has written the Programming Language Annex for the SAE AADL standard. Dr. Tokar has also participated in the evolution of the Ada programming language both as a member of the team defining the Ada 05 update and as a distinguished reviewer for Ada 95.

Dr. Tokar is also active in the area of secure software system development tools and environments. She is leading a team in the analysis and evolution of the system of systems software for the US Department of Defense Future Combat System (FCS).

From 1981-84 Dr. Tokar was responsible for the development of the Gensoft (Western Digital) Ada system. She received her PhD in Computer Engineering from Clemson University in South Carolina. She holds an MS and a BS in Computer Science from the University of Pittsburgh.

The Ada Paradox(es)

Jean-Pierre Rosen, *President, Adalog*

Jean-Pierre Rosen graduated from ENST (Ecole Nationale Supérieure des Télécommunications) in 1975, and attained the PhD in 1986. He started as a software engineer at the computing center of ENST. After a Sabbatical at New York University on the Ada/ED Project, he worked as Professor at ENST, where he was responsible for the teaching of Software Engineering and Ada.

He has now formed Adalog, a company specialized in high level training, consultancy, and software development in the fields of Ada and associated technologies (software engineering, object oriented methodologies).

J-P. Rosen is Chairman of the AFNOR (French standardization body) group for Ada, and a member of the ARG (Ada Rapporteur Group), the group of experts in charge of maintenance and evolution of the language. He was a member of the expert team who controlled the development of the validation suite for Ada 95.

He is the author of "Méthodes de Génie Logiciel avec Ada 95" (Software Engineering Methods with Ada 95) and "HOOD: an industrial approach for software development".

Call for Papers

14th International Conference on Reliable Software Technologies – Ada-Europe 2009

8-12 June 2009, Brest, France

http://www.ada-europe.org/conference2009.html

General Information

The 14th International Conference on Reliable Software Technologies – Ada-Europe 2009 will take place in Brest, France. Following its traditional style, the conference will span a full week, including a three-day technical program and vendor exhibitions from Tuesday to Thursday, along with parallel tutorials and workshops on Monday and Friday.

Schedule

01 December 2008	Submission of regular papers, tutorial and workshop proposals
12 January 2009	Submission of industrial presentation proposals
09 February 2009	Notification to all authors
09 March 2009	Camera-ready version of regular papers required
11 May 2009	Industrial presentations, tutorial and workshop material required
08-12 June 2009	Conference

Topics

The conference has successfully established itself as an international forum for providers, practitioners and researchers into reliable software technologies. The conference presentations will illustrate current work in the theory and practice of the design, development and maintenance of long-lived, high-quality software systems for a variety of application domains. The program will allow ample time for keynotes, Q&A sessions, panel discussions and social events. Participants will include practitioners and researchers in representation from industry, academia and government organizations active in the promotion and development of reliable software technologies. To mark the completion of the Ada language standard revision process, contributions that present and discuss the potential of the revised language are particularly sought after.

Prospective contributions should address the topics of interest to the conference, which include but are not limited to those listed below:

- **Methods and Techniques for Software Development and Maintenance**: Requirements Engineering, Object-Oriented Technologies, Model-driven Architecture and Engineering, Formal Methods, Re-engineering and Reverse Engineering, Reuse, Software Management Issues, Model Engineering.
- **Software Architectures**: Design Patterns, Frameworks, Architecture-Centered Development, Component and Class Libraries, Component-based Design.
- **Enabling Technologies**: Software Development Environments and Project Browsers, Compilers, Debuggers, Run-time Systems, Middleware Components.
- **Software Quality:** Quality Management and Assurance, Risk Analysis, Program Analysis, Verification, Validation, Testing of Software Systems.
- **Theory and Practice of High-integrity Systems**: Real-Time, Distribution, Fault Tolerance, Security, Reliability, Trust and Safety.
- **Embedded Systems**: Architecture Modeling, Co-Design, Reliability and Performance Analysis.
- **Mainstream and Emerging Applications**: Multimedia and Communications, Manufacturing, Robotics, Avionics, Space, Health Care, Transportation.
- **Ada Language and Technology**: Programming Techniques, Object-Orientation, Concurrent and Distributed Programming, Evaluation & Comparative Assessments, Critical Review of Language Features and Enhancements, Novel Support Technology, HW/SW Platforms.
- **Experience Reports**: Case Studies and Comparative Assessments, Management Approaches, Qualitative and Quantitative Metrics.
- **Ada and Education**: Where does Ada stand in the software engineering curriculum; how learning Ada serves the curriculum; what it takes to form a fluent Ada user; lessons learned on Education and Training Activities with bearing on any of the conference topics.

Call for Regular Papers

Authors of regular papers which are to undergo peer review for acceptance are invited to submit original contributions. Paper submissions shall be in English, complete and not exceeding 14 LNCS-style pages in length. Authors should submit their work via the Web submission system accessible from the Conference Home page. The format for submission is solely PDF. Should you have problems to comply with format and submission requirements, please contact the *Program Chair*.

Proceedings

The authors of accepted regular papers shall prepare camera-ready submissions in full conformance with the LNCS style, not exceeding 14 pages and strictly by 9 March 2009. For format and style guidelines authors should refer to: http://www.springer.de/comp/lncs/authors.html. Failure to comply and to register for the conference will prevent the paper from appearing in the proceedings. The conference proceedings will be published in the Lecture Notes in Computer Science (LNCS) series by Springer Verlag, and will be available at the start of the conference.

Awards

Ada-Europe will offer honorary awards for the best regular paper and the best presentation.

Call for Industrial Presentations

The conference also seeks industrial presentations which may deliver value and insight, but do not fit the selection process for regular papers. Authors of industrial presentations are invited to submit a short overview (at least 1 page in size) of the proposed presentation to the *Conference Chair* by 12 January 2009. The *Industrial Program Committee* will review the proposals and make the selection. The authors of selected presentations shall prepare a final short abstract and submit it to the *Conference Chair* by 11 May 2009, aiming at a 20-minute talk. The authors of accepted presentations will be invited to derive articles from them for publication in the Ada User Journal, which will host the proceedings of the Industrial Program of the Conference.

Call for Tutorials

Tutorials should address subjects that fall within the scope of the conference and may be proposed as either half- or full-day events. Proposals should include a title, an abstract, a description of the topic, a detailed outline of the presentation, a description of the presenter's lecturing expertise in general and with the proposed topic in particular, the proposed duration (half day or full day), the intended level of the tutorial (introductory, intermediate, or advanced), the recommended audience experience and background, and a statement of the reasons for attending. Proposals should be submitted by e-mail to the *Tutorial Chair*. The providers of full-day tutorials will receive a complimentary conference registration as well as a fee for every paying participant in excess of 5; for half-day tutorials, these benefits will be accordingly halved. The Ada User Journal will offer space for the publication of summaries of the accepted tutorials.

Call for Workshops

Workshops on themes that fall within the conference scope may be proposed. Proposals may be submitted for half- or full-day events, to be scheduled on either ends of the conference week. Workshop proposals should be submitted to the *Conference Chair*. The workshop organizer shall also commit to preparing proceedings for timely publication in the Ada User Journal.

Call for Exhibitions

Commercial exhibitions will span the three days of the main conference. Vendors and providers of software products and services should contact the *Exhibition Chair* for information and for allowing suitable planning of the exhibition space and time.

Grants for Students

A limited number of sponsored grants is expected to be available for students who would like to attend the conference or tutorials. Contact the *Conference Chair* for details.

In cooperation with SIGAda
(*approval pending*)

13th International Real-Time Ada Workshop

17-19 April 2007
Woodstock, Vermont
USA

Sessions:
Implementation Experience with Ada 2005
Beyond Ada 2005

from the Proceedings* edited by: Juan Antonio de la Puente

Program Committee

Alan Burns	Javier Miranda	José F. Ruiz
Ben Brosgol [b]	Luis Miguel Pinho	Tullio Vardanega
Michael González Harbour	Juan Antonio de la Puente [a]	Andy Wellings
Stephen Michell	Jorge Real	

[a] Program Chair [b] Local Chair

Workshop Participants

Name	Institution
Mario Aldea Rivas	Universidad de Cantabria, Spain
Neil Audsley	University of York, UK
Ben Brosgol	AdaCore, USA
Alan Burns	University of York, UK
Michael González-Harbour	Universidad de Cantabria, Spain
J. Javier Gutiérrez	Universidad de Cantabria, Spain
Stephen Michell	Maurya Systems, Canada
Brad Moore	General Dynamics, Canada
Juan Antonio de la Puente	Universidad Politécnica de Madrid (UPM), Spain
Jorge Real	Universidad Politécnica de Valencia, Spain
José F. Ruiz	AdaCore, France
J.C. Smart	Department of Defense, USA
Santiago Urueña	Universidad Politécnica de Madrid (UPM), Spain
Tullio Vardanega	University of Padua , Italy
Andy Wellings	University of York, UK
Rod White	MBDA, UK
Curtis Winters	Aonix, USA
Juan Zamorano	Universidad Politécnica de Madrid (UPM), Spain

Sponsors

* The complete Proceedings of the 13th International Real-Time Ada Workshop previously appeared in ACM Ada Letters, Volume XXVII, Number 2, August 2007; reprinted with permission.

Session: Implementation Experience with Ada 2005

Chair: Alan Burns
Rapporteur: Andy Wellings

Session Goals

The goals of this session were to

- Discuss implementation experience with the new real-time features
- Review the support provided by the new real-time features
- Review features proposed but omitted from Ada 2005

Real-Time Utilities

Mario Aldea Rivas first gave an overview of the approach their paper had taken on implementing the new Ada 2005 real-time services in MaRTE OS and GNAT. The most important of these new services are:

- timing events
- execution-time clocks and timers
- dynamic priorities for protected objects
- immediate priority changes
- group execution-time budgets
- new scheduling and task dispatching mechanisms

Of these, the first five had already been implemented and would be released by AdaCore in the near future. The rest would be done during the summer of 2007.

For timing events, Mario indicated that it was not possible to implement timing event straight from the clock interrupt handler as there was no mechanisms provided by POSIX to do so. He indicated that there were essentially two approaches: one where run-time threads are introduced for each timing event, the other where the OS is changed.

He said they had implemented both approaches and that by changing the OS, there was a significant performance gain. For execution-time clocks and handlers, Mario reported that the implementation was much simpler as both the POSIX and Ada 2005 standards took a similar stance. In particular:

- Neither of the standards define which task/thread is charged with the overheads of interrupt handlers and run-time services on behalf of the system
- Both standards state that the execution time is set to zero at the creation of the task /thread
- Ada 2005 says the time spent during task activation must be charged to the task execution time clock – this happens in GNAT since activation is executed by the thread used to implement the Ada task.

As a consequence, no modifications to the compiler or to the run-time system have been necessary. Mario reported that execution time accounting introduces a small overhead to context switch time (less than 5%) and that the time to read execution time clocks is very similar to the time to read the real-time clock.

Execution-timers had been built on top of the timers and had caused no significant implementation problems. Group execution time accounting, however, required significant modifications to the OS as POSIX did not support thread groups. The facility added an extra 9% overhead on context switch times.

Juan Zamorano gave a presentation on their implementation of the same facilities in the Open Ravenscar Kernel on a bare board Leon (based on the SPARC V8 architecture). Juan indicated that the scarce hardware support for timers on that board meant that significant software support was required. This had added a 50% increase in context switch times. The full details are given in the paper.

Following the two presentations it was noted that the Workshop was not aware of other projects implementing 2005 real-time facilities.

Discussions on the New Features

The main discussions following the presentations focused on the overheads and inaccuracies of the CPU accounting model.

The following issues were raised:

1. Context switch time – Mario reported that there was no leakage of CPU time during context switches.
2. System and Application interrupts (e.g. clocks) – whil-st Ada allowed interrupt handling to be charged to the executing tasks there was concerns that this was a significant inaccuracy.
3. Timing events code – it was again noted that the code executed by timing event handlers was application level code and therefore was not fixed. This would again be charged inappropriately to the running task.

This paper previously appeared in ACM Ada Letters, Volume XXVII, Number 2, August 2007; reprinted with permission.

However, it was also pointed out that as the code was a protected procedure, the time was at least bounded per handler.

4. Proxy model of Protected Objects – concern was expressed that the proxy model of implementation for protected objects could result in a significant inaccuracy as one task could execute a significant amount of code on behalf of another.

There was a long discussion of whether the CPU accounting model was useable given the inherent inaccuracies. Various points were noted:

- The facilities could be used with a measurement-based approach. Execution time could be measured during system testing and this figure used at run-time. However, this approach is fragile. Any small change to the application code would mean that the system-level timing measurements would have to be redone.
- For hard real-time systems, it was noted that there had to be an associated analysis model. The worst-case overheads could then be added to the execution time of each task. However, this approach could be very pessimistic as each task would be charged the worst case overhead.
- It was also pointed out that the greatest error was on the value of the worst-case execution time itself and that adding a small error was at the noise level.

Another point raised was that the impact of handling low priority interrupts on high priority task could be significant.

The workshop concluded that there is a need to investigate the overheads and the extra cost of trying to do better accounting. Also the overhead of a better model of prioritized interrupt handling should be investigated.

Application-level Scheduling

Michael Gonzalez Harbour gave an overview of the current status of application-defined scheduling work that had been reported at the last workshop. Although this had failed to get in to the standard, an implementation had been produced and would be released as an extension to GNAT. The hope was that people would use the facilities and that it might eventually become a de facto standard. The workshop reaffirmed its support for the need of such a facility in Ada.

Ravenscar

This session of the Workshop concluded with a discussion of the continuing experience with the Ravenscar profile. Juan Antonio de la Puente raised the issue of execution timers and group budgets. Although Ravenscar allows execution-time clocks, it prohibits timers and group budgets. He proposed that we should allow one timer per task. The motivation is to make sure a task does not consume more than its budget.

Whilst there was some support for this proposal, concern was expressed on how a Ravenscar program would respond to a timer expiring. There are not asynchronous interaction mechanisms in Ravenscar. Juan Antonio indicated that this was similar to the way task termination was handled. If a task terminated in Ravenscar (which it should not), the event is brought to the attention of the program and then it is implementation-defined what mechanisms the programmer can use.

It was pointed out that a monitor task could always read the execution times of other tasks and discover the overrun. However, there would clearly be a delay in doing this. There was no consensus position reached.

The Workshop felt that adding Group budgets opened up a new profile. This ought to be considered perhaps in a context where there are more than one Ravenscar applications (in effect, partitions) running on the same run-time.

Summary

The following summarised the positions taken by the Workshop during this session:

1. There is a need to investigate the overheads and the extra cost of trying to do better accounting and of the overheads of doing a better model of prioritized interrupt handling.
2. There is continued support for application-defined scheduling.
3. There is no consensus on adding CPU Timers into Ravenscar (i.e. it is an open issue that needs further investigation).
4. Group budgets and the coexistence of multiple Ravenscar applications on a single processing node needs further investigation.

Implementing the New Ada 2005 Real-Time Features on a Bare Board Kernel*

S. Urueña, J. Pulido, J. Redondo, J. Zamorano

Universidad Politécnica de Madrid (UPM), Spain; email: {suruena@datsi.fi, pulido@dit, jredondoh@dit, jzamora@datsi.fi}.upm.es

Abstract

A real-time kernel providing timing services is a key component of any real-time system. The current revision of the Ada standard provides a range of real-time mechanisms that can be used to ensure the required temporal behaviour of real-time tasks. However, kernel timing services must be implemented carefully in order to avoid overheads and inaccuracies. This paper describes the implementtation of the Ada timing services in an evolved version of the Open Ravenscar Kernel. The interrelation among the different timing mechanisms is also analysed and evaluated.

1 Introduction

High-integrity real-time systems usually have hard timing requirements, which have to be guaranteed by using an appropriate engineering approach for their design and implementation (see e.g. [19]). Such an approach is usually based on a computation model which enables the temporal behaviour of a system to be analysed and adjusted if necessary.

Ensuring the required real-time behaviour usually relies on an accurate knowledge of the worst-case computation times (WCET) of all the real-time tasks. Although some good techniques for computing WCET are available [16], there is still a large degree of uncertainty, especially when modern processors with cache memories, speculative execution and segmentation are used. Pessimistic WCET estimations lead to an underutilisation of resources, and thus tight estimates are usually sought. The risk with tight WCET estimates is, on the other hand, to be optimistic, and then occasionally get an actual execution time which is larger than the estimated value. This situation is called an overrun, and may give raise to a generalised miss of deadlines by tasks by a domino effect.

The new Ada real-time mechanisms can be used to monitor the run-time behaviour of tasks. In this way it is pos sible to detect overruns and deadline misses and take corrective actions before other tasks are affected [14]. This paper presents the implementation of the Ada real-time features on GNATforLEON, an open-source cross-compilation system that implements Ravenscar tasking for LEON2 [10] targets. GNATforLEON is a port to LEON2 targets of GNAT Pro for ERC32 [17]. GNAT Pro for ERC32 and thus GNATforLEON uses a version of the GNAT run-time library (GNARL) specially developed to support the Ravenscar profile on top of a bare board kernel which is an evolved version of the Open Ravenscar Kernel (ORK) [5, 7].

2 The new Ada 2005 real-time mechanisms

The current revision of the Ada standard provides a range of real-time mechanisms that can be used to ensure the required temporal behaviour of real-time tasks. The Ada.Real_Time package includes a monotonic real-time Clock as well as a definition of Time which are appropriate for real-time systems. The package was already part of Ada 95 [12, Annex D] and can be used to check real-time related properties, such as minimum inter-arrival times or task deadlines. Real-time timers were not provided as such in Ada 95, but delay statements and asynchronous transfer of control (ATC) provided a similar functionality at a higher abstraction level (see e.g. [4]). However, ATC is excluded from the Ravenscar profile due to its complex implementation. Nevertheless, there are new real-time mechanisms which can be used to efficiently detect deadline overruns in critical systems.

Timing events [1] is an Ada 2005 lower-level mechanism that can be used with the Ravenscar profile [18, D.15] for detecting deadline overruns [14]. Timing events are a light-weight mechanism for specifying an action to be executed at a given time without the need to use a task or a delay statement. A timing event can be set to occur at an absolute time or after a real-time interval. A protected procedure handler is executed whenever the event occurs, unless it is cancelled before that time. The functionalityof timing events is provided by the library-level package Ada.Real_Time.Timing_Events, in this way it is no needed to change the compiler to implement this mechanism but just to add the support to the Ada run-time library as well as to the underlying kernel. It is worth noting that only library-level timing events are allowed by the Ravenscar profile.

Ada 2005 also includes mechanisms for measuring and monitoring execution-time, namely *execution-time clocks*

* This work has been partially funded by the IST Programme of the European Commission under project IST-004033 (ASSERT) and the Spanish Ministry of Science and Technology (MCYT), project TICTIC2005-08665-C03-01 (THREAD).

This paper previously appeared in ACM Ada Letters, Volume XXVII, Number 2, August 2007; reprinted with permission.

and *timers* [2], and *group execution-time budgets* [3]. These mechanisms can be used to estimate the execution time of code segments, to handle some kinds of aperiodic events, and to detect execution-time related temporal faults. These mechanisms are also provided by library-level packages: Ada.Execution_Time, Ada.Execution_Time.Timers, and Ada.Execution_Time.Group_Budgets.

In Ada 2005 each task has an *execution-time clock* that computes the amount of CPU time it has consumed, including the run-time services invoked by the task. It should be noticed that it is implementation-defined which task is charged with the execution time for system services, which include interrupt service routines, or even whether it is charged to no task [18, D.14(13a/2)]. *Execution-time timers* are objects that are associated with a task —and hence with the task execution-time clock— when they are declared. A timer can be armed to expire at an absolute value of that clock or after some execution-time interval. When the timer expires, a protected procedure handler is executed. Setting again the handler replaces the handler and the time of execution and the timer remains set. *Group execution-time budgets* is a similar mechanism, which can be used with a set of tasks instead of a single task. A task can belong to at most one such group. A global budget of execution-time can be allocated to the whole group, and then it is decreased when any task in the group consumes execution time. As with timers, a protected procedure handler can be specified to be executed whenever the budget is exhausted. The budget can also be replenished at any time.

Execution-time clocks are allowed in the Ravenscar profile, but timers and group budgets are not. However, we believe that these mechanisms can be safely and efficiently used in high-integrity systems, provided that they are only declared at library level and there is at most one execution-time timer per task [6].

3 Kernel support for timing services

The described timing services has been implemented on GNATforLEON which is an evolved version of ORK for LEON2 based computers. LEON2 is a radiation-hardened implementation of the SPARC V8 architecture, which has been adopted by the European Space Agency (ESA) as the new standard processor for spacecraft on-board computer systems as an upgrade of the ERC32 [9].

GNATforLEON provides direct support for the Ravenscar profile [18, D.13.1], including the following Ada 2005 timing services:

- Global timing events;
- Execution-time clocks.

Execution-time timers and group budgets are also supported by the kernel in spite of being not allowed by the Ravenscar profile. These mechanisms are needed to enforce temporal separation in logical partitioned systems where subsystems with possibly different levels of criticality can share computer nodes. This is a strong requirement for the kind of on-board aerospace embedded systems envisaged in the ASSERT project[1][15].

The implementation of Ada.Real_Time.Clock and absolute delays for ORK/ERC32 is thoroughly described in [21]. It is based on the two 32-bit hardware timers of the ERC32 processor. That implementation has been ported to the LEON2 processor which has two 24-bit hardware timers. It is worth noting that Annex D of the Ada Language Reference Manual [18] requirements for Ada.Real_Time.Time lead to at least 41 bits for that type. As a result, the implementation uses the hardware timer register as the least significant part (LSP) of the clock and a 32-bit word in memory as the most significant part (MSP). This arrangement provides an accurate tick with low overhead.

Execution-time clocks and timers were also supported by ORK/ERC32 whose implementation is described in [20]. The implementation only allows one execution-time timer per task, as suggested in previous IRTAW discussions [8, 6] and permitted by the Ada Language Reference Manual [18, D.14.1(28/2)]. Although execution-time timers are not allowed in the Ravenscar profile, the Ravenscar profile restrictions enable a simple and efficient implementation which was ported to GNATforLEON. Group budgets were not implemented in ORK but recently on GNATforLEON. However, the implementation is built on top of the execution-time timers one and thus only little support is needed for group budgets at kernel level. The main part is at Ada run-time level in the body of the Ada.Execution_Time.Group_Budgets whereas at kernel level is only needed a flag to record if the armed execution-time alarm of the task correspond to its execution-time timer or to the group budget timer at which the task belongs. In this way, the proper handler may be invoked if the execution-time alarm expires.

The overall implementation is schematically shown in figure 1. As said, timer 1 is used in periodic mode to support Ada.Real_Time.Clock and timer 2 is used in one-shot mode and is armed to expire with the closest event. This event can be an absolute delay or the execution-time timer of the running task, which in turn could be its own timer or the timer of its group budget. In a similar way to real-time clock, the execution-time clock of the running tasks is built up by using the hardware timer register and the cumulated execution time. However, if the timer 2 is armed with an absolute delay is more complex to build up the execution-time clock of the running tasks.

4 Implementation of timing-events

The Ravenscar profile restrictions avoids delay cancellation and therefore the alarm queue of figure 1 is simply linked. However timing-events can be cancelled and the alarm queue can not be efficiently used for this purpose.

[1] ASSERT (Automated proof based System and Software Engineering for Real-Time) is an FP6 Integrated Project coordinated by the European Space Agency. The main goal of the project is to improve the system-andsoftware development process for critical embedded real-time systems, in the Aerospace and Transportation domains

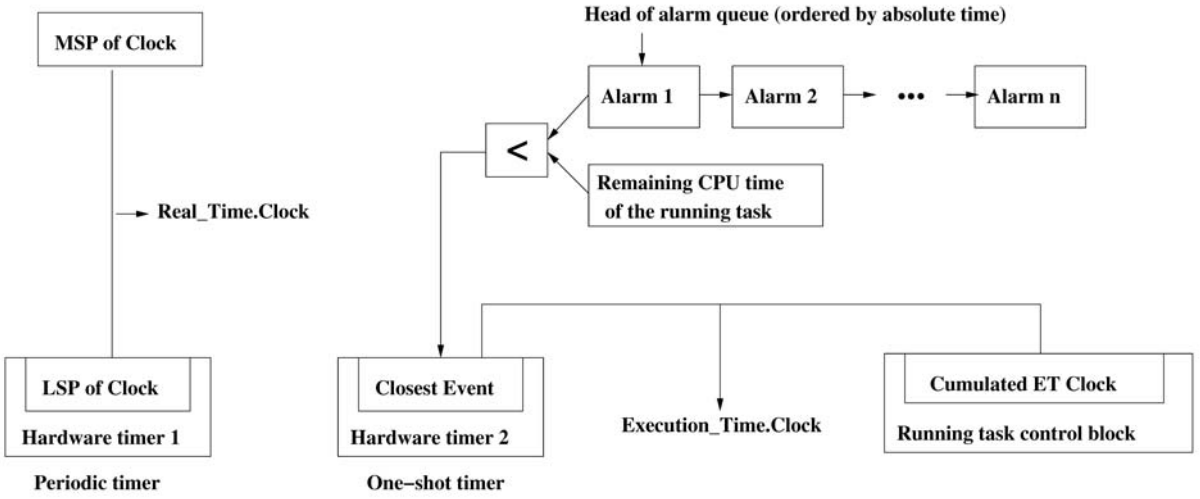

Figure 1 Schema of clocks and timers implementation

Nevertheless it is possible to manage timing-events in the same way of absolute delays, that is by inserting them in the simply linked alarm queue and if a timing event is cancelled its associate handler is set to null.

The overhead of the above approach could be intolerable in applications with timing event cancellations. It should be noticed that the processing of the timer interrupts implies the execution of the preamble and the epilogue together with the run-time alarm handler. This alarm handler has to clear the interrupt, identify the type of the event, jump to an Ada code which is the handler and finally look for the next closest event in order to arm the hardware timer. In our opinion, this is a pretty amount of instructions for a null handler.

As a result, it was decided to use a new doubly linked queue for timing events. Therefore, timing events can be simply and efficiently located and removed when they are cancelled. The timing event queue is also ordered by absolute time in spite of the timing event was set by using relative time, as it is the best approach [21].

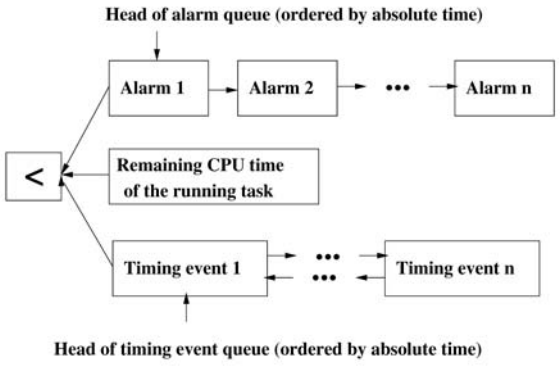

Figure 2 Schema of event queues.

The figure 2 shows the new queues arrangement. In general, it is more simple and efficient to maintain two queues than the combined one and thus the implementation has a lower overhead. It could be argued that it is needed to compare among three events in order to identify the closest event and therefore more comparisons are made in order to arm the hardware timer than without a new queue. This is a fallacy because the total number of comparisons is even lower because it is made much more comparisons for inserting the event in a longer queue.Finally, it must be said that the implementation could be much more simple with a little bit of hardware support. For instance, the implementation of real-time clock and absolute delays of the ORK version for PC computers is much more efficient and simple because it takes advantage of the Time Stamp Counter which can be found in Pentium processors. The Time Stamp Counter is an up-count 64-bit timer and thus it is able to maintain the monotonic real-time clock itself. This is fairly convenient making more simple to operate the alarm queue because reading the clock is just one instruction.

5 Implementation inaccuracies

The described implementation allocates the time spent in interrupt service routines to the currently running task, which is allowed by Ada. However, it allocates the time spent in timing event handlers too. It should be noticed that to stop and restart the execution-time clock of the running task is not easy because it is not just a matter of stopping the hardware timer. As it is shown in figure 2, the hardware timer 2 does not hold the remaining CPU time of the running task when a timing event expires but the expired timing event. As a result, the real-time clock should be recorded at the beginning and the end of the hardware timer 2 interrupt handler in order to properly updated the remaining CPU time of the running task by subtracting this elapsed time.

In this way, it is not only complex to avoid the allocation of the time spent in timing event handlers to the running task but the time spent for avoiding this could be greater than the time spent in the timing event handler itself. Indeed, it

is a time-spending operation to read the clock with the little hardware support of LEON2 processors.

Nevertheless, we believe that the implementation is implicitly allowed by the standard because the Ada Language Reference Manual [18, D.15(25/2)] says that "The protected handler procedure should be executed directly by the real-time clock interrupt mechanism". As it is implementation-defined which task is charged execution time for the time spent in interrupt service routines, it can be concluded that it is also allowed to charge the time spent in timing event handlers. In our honest opinion, it should be clarified in an Ada Issue.

Another source of inaccuracy of execution-time clocks is the so-called *proxy model* for servicing the entries of protected objects which is used by the GNAT compiler [13]. With this approach the task exiting the eggshell executes all the waiting entry calls whose barriers are open on behalf the awaiting tasks and reevaluates the barriers every time. As a result, this time spent for other tasks is charged to the execution-time clock of the exiting task.

The Ravenscar profile only allows one awaiting task per protected object and therefore this inaccuracy can be bounded but as there is no language-imposed restriction on the number of such calls that can be pending, the inaccuracy could be intolerable for general Ada programs. A way to avoid this inaccuracy could be to use the so-called *self- service model*, although the number of context switchings would increase. Moreover, the GNAT compiler and the GNAT Ada run-time library should be modified in order to do that.

6 Metrics

The Ada 2005 timing mechanisms have been implemented by the authors on GNATforLEON, a compilation system for the LEON2 processor, a radiation-hardened derivative of the SPARCv8 RISC architecture for the space domain. The implementation has been based on a previous experimental implementation on top of the Open Ravenscar kernel [20]. The modified compilation system is being used as the execution platform for the ASSERT project.

The overhead of the new timing mechanisms (execution-time timers, group budgets, and timing events) has been measured by comparing footprint size and context switch duration between GNATforLEON 1.0 and GNATforLEON 1.3. GNATforLEON 1.0 is the first version of the compilation system which does not have the new timing mechanisms. Conversely GNATforLEON 1.3 includes all of them. The values shown in tables 1 and 2 have been measured using a pilot application, and therefore should be considered as average values, not as worst-case metrics.

Table 1 Context switch in GNATforLEON

Run-time system	**Context switch** (instructions)
GNATforLEON 1.0	405
GNATforLEON 1.3	606

Table 2 Memory footprint

Section	**Size** (kilobytes)	
	GNATforLEON 1.0	GNATforLEON 1.3
. text	79	87
.data	8	8
.bss	362	365
Total	449	460

The overhead is moderate as about 200 new instructions have to be executed per context switching to support execution time clock and group budget on GNATforLEON. The absolute timing impact depends highly on actual CPI (Cycles Per Instruction) which in turns depends on the status of pipeline, caches and register window. The ideal CPI is 1 and the clock frequency of LEON2 processor is 50 MHz therefore the minimum absolute overhead is 4 μs.

Table 3 shows the instructions required for timing service primitives. It should be noticed that 71 instructions are just needed to read the clock and the implementation needs to read the clock during context switching and timing service primitives. As a result, the poor hardware support of LEON2 processors highly impact on the duration of timing service primitives.

Table 3 Primitives

Operation	**Instructions**
Real_Time.Clock	71
Timing_Events.Set_Handler	240
Execution_Time.Timers.Set_Handler	271

Table 4 shows the latencies for executing the corresponding handler when an execution time or group budget timer expires. 400 instructions are needed from the first instruction of the low-level interrupt handler to the first instruction of the Ada handler. It is fairly low providing that LEON2 is a RISC processor.

Table 4 Handler latencies

Operation	**Instructions**
Timing event handler	396
Execution-time handler	415

The footprint increases in 11 kbytes which are mainly due to the 8 kbytes augment in the code (text section). The other 3 kbytes are due to the need of larger ATCB and structures for individual objects. Table 5 shows the footprint of the required structures.

Table 5 Memory size

Type	**Size** (bytes)
Timing_Event	24
Execution_Time.Timers.Timer	20
Group_Budget	2064

7 Hardware support

The overhead introduced by real-time mechanisms in the kernel primitives is moderate. However, as said above, 71 instructions are needed to read the clock and the clock must be read to obtain the relative down-count that should be loaded in the hardware register. In this way, the fourth part of the extra instructions in a context switching are used just for reading the clock. There are also another operations which are time consuming such as to compare 64-bit time values and to convert a relative time in the corresponding value that should be load in the down-count timer register.

As a result, a significant part of the introduced overhead can be avoided with just a little bit of hardware support. Hardware timers are fairly simple devices and they can be included in a processor board at a very low cost. It can be envisage a very simple implementation of the described real-time mechanisms just with four 64-bit up-count hardware timers.

In this way, it can be used one timer to support the monotonic real-time clock on hardware, as with the Pentium Time Stamp Counter, without any software support. A second one can be used for the absolute delay queue which is ordered by absolute time and thus the absolute expiration time would be loaded in the so-called *c*omparator value Register of the hardware timer. Therefore, an interrupt request should be delivered when the up-count timer reaches the comparator value. A separate timer can be used for timing events which can be managed with the same approach although the timing event queue should be doubly linked. The last timer is dedicated to count for the execution time of the running task, in this way it would be easy read the execution-time clock of the running task. Moreover, to stop and restart this timer would be the needed simple operations to avoid charging the time spent in interrupt service routines and timing events to the running task.

Recently, Intel has specified the so-called High Precision Event Timers [11] for the PC architecture. The specification defines a block of up-count 64-bit timers and each timer can be configured to generate a separate interrupt. The specification allows for a block of 32 timers, with support for up to 8 blocks, which allows a total of 256 timers.

The specification fulfils the requirements to implement the real-time mechanism with a low overhead because timers are implemented as a single up-counter with a set of comparators. Each timer includes a match register and a comparator, and can generate an interrupt when the value in its match register equals the value of the free-running counter. Moreover, the counters increases monotonically and some of the timers can be enabled to generate a periodic interrupt.

It can be easily envisage a very simple implementation of the real-time services with such population of timers. Every timing event could use its own hardware timer and thus queueing is avoided. However, it should be needed to limit the maximum number of timing events with the corresponding pragma Restrictions. In a similar way, every task could own a hardware timer in order to support its execution-time clock and timer. As a result, the overhead in context switching would be reduced to stop and restart the corresponding timers of both tasks. It should be noticed that the maximum number of tasks can be limited by a pragma Restrictions and the implementation may limit the number of timers that can be defined for each task to one, and thus this implementation is allowed by the standard.

Finally, it could be possible to use a periodic timer to activate each periodic task and to eliminate the alarm queue too. Unfortunately, Ada has not a way of specifying the period of a real-time periodic task and it would be needed to add this feature by a specific implementation pragma.

8 Conclusions

The Ada 2005 real-time services are of paramount importance for detecting temporal faults and thus they enable the development of fault tolerant systems. The implementation described in this paper has a moderate overhead for a Ravenscar kernel and does not introduce much complexity to the underlying kernel. Therefore, they can be used for building high integrity systems.

It should be noticed that the hardware timer devices of LEON2 processor are not adequate to support the real-time features which are needed in a real-time system. Even the monotonic real-time clock needs a significant software support. We believe that the overhead can be highly reduced with a little bit of hardware support which can be found in the Intel PC architecture.

Additionally, some inaccuracies in the implementation of execution-time timers are derived from this poor hardware support, as well as due to the proxy model. These inaccuracies can not be completely avoided in the general case and implementation advices should be provided.

References

[1] Ada Rapporteur Group. Ada Issue 297 — Timing events. *Ada Letters*, XXV(3), September 2005.

[2] Ada Rapporteur Group. Ada Issue 307 — Execution-time clocks. *Ada Letters*, XXVI(1), April 2006.

[3] Ada Rapporteur Group. Ada Issue 354 — Group execution-time budgets. *Ada Letters*, XXVI(2), August 2006.

[4] A. Burns and A. J. Wellings. *Real-Time Systems and Programming Languages*. Addison-Wesley, 3 edition, 2001.

[5] J. A. de la Puente, J. F. Ruiz, and J. Zamorano. An open Ravenscar real-time kernel for GNAT. In H. B. Keller and E. Plödereder, editors, *Reliable Software Technologies — Ada-Europe 2000*, number 1845 in LNCS, pages 5–15. Springer-Verlag, 2000.

[6] J. A. de la Puente and J. Zamorano. Execution-time clocks and Ravenscar kernels. *Ada Letters*, XXIII(4):82–86, December 2003. Proceedings of the

12th International Ada Real-Time Workshop (IRTAW12).

[7] J. A. de la Puente, J. Zamorano, J. F. Ruiz, R. Fernández, and R. García. The design and implementation of the Open Ravenscar Kernel. *Ada Letters*, XXI(1), 2001.

[8] B. Dobbing and J. A. de la Puente. Session report: Status and future of the Ravenscar profile. *Ada Letters*, XXIII(4):55–57, December 2003. Proceedings of the 12th International Real-Time Ada Workshop (IRTAW 12).

[9] ESA. *32 Bit Microprocessor and Computer System Development*, 1992. Report 9848/92/NL/FM.

[10] GR. *LEON2 Processor User's Manual*, 2005. Gaisler Research.

[11] Intel Corporation. *IA-PC HPET (High Precision Event Timers) Specification*, 2004. Intel Corporation.

[12] *Ada 95 Reference Manual: Language and Standard Libraries. International Standard ANSI/ISO/IEC-8652:1995*, 1995. Available from Springer-Verlag, LNCS no. 1246.

[13] J. Miranda. *A Detailed Description of the GNU Ada Run Time*. http://www.iuma.ulpgc.es/users/jmiranda/gnat-rts/, 2003.

[14] J. A. Pulido, S. Urueña, J. Zamorano, and J. A. de la Puente. Handling temporal faults in Ada 2005. In N. Abdennadher and F. Kordon, editors, *Reliable Software Technologies — Ada-Europe 2007*, number 4498 in LNCS, pages 15–28. Springer-Verlag, 2007.

[15] J. A. Pulido, S. Urueña, J. Zamorano, T. Vardanega, and J. A. de la Puente. Hierarchical scheduling with Ada 2005. In L. M. Pinho and M. González Harbour, editors, *Reliable Software Technologies — Ada-Europe 2006*, volume 4006 of *LNCS*. Springer Berlin / Heidelberg, 2006.

[16] P. Puschner and A. Burns. A review of worst-case execution time analysis. *Real-Time Systems*, 18(2/3):115–128, May 2000.

[17] J. F. Ruiz. GNAT Pro for on-board mission-critical space applications. In T. Vardanega and A. Wellings, editors, *Reliable Software Technologies — Ada-Europe 2005*, volume 3555 of *LNCS*. Springer-Verlag, 2005.

[18] S. T. Taft, R. A. Duff, R. L. Brukardt, E. Ploedereder, and P. Leroy, editors. *Ada 2005 Reference Manual. Language and Standard Libraries. International Standard ISO/IEC 8652/1995(E) with Technical Corrigendum 1 and Amendment 1*. Number 4348 in Lecture Notes in Computer Science. Springer-Verlag, 2006.

[19] T. Vardanega. Development of on-board embedded real-time systems: An engineering approach. Technical Report ESA STR-260, European Space Agency, 1999.

[20] J. Zamorano, A. Alonso, J. A. Pulido, and J. A. de la Puente. Implementing execution-time clocks for the Ada Ravenscar profile. In A. Llamosí and A. Strohmeier, editors, *Reliable Software Technologies — Ada-Europe 2004*, volume 3063 of *LNCS*. Springer-Verlag, 2004.

[21] J. Zamorano, J. F. Ruiz, and J. A. de la Puente. Implementing Ada.Real_Time.Clock and absolute delays in real-time kernels. In A. Strohmeier and D. Craeynest, editors, *Reliable Software Technologies — Ada-Europe 2001*, number 2043 in LNCS, pages 317–327. Springer-Verlag, 2001.

Operating System Support for Execution Time Budgets for Thread Groups

Mario Aldea Rivas, Michael González Harbour
Universidad de Cantabria, 39005-Santander, Spain; email:{mgh, aldeam}@unican.es

Abstract

The recent Ada 2005 standard introduced a number of new real-time services, with the capability of creating and managing execution time budgets for groups of tasks. This capability has many practical applications in real-time systems in general, and therefore it is also interesting for real-time operating systems. In this paper we present an implementation of thread group budgets inside a POSIX real-operating system, which can be used to implement the new Ada 2005 services. The architecture and details of the implementation are shown, as they may be useful to other implementers of this functionality defined in the new standard.

Keywords: Real-time systems, Execution time budgets, Thread groups, CPU time, Ada 2005.

1 Introduction[1]

In hard real-time systems it is essential to monitor the execution times of all tasks and detect situations in which the estimated worst-case execution time (WCET) is exceeded. This detection was usually available in systems scheduled with cyclic executives, because the periodic nature of its cycle allowed checking that all initiated work had been completed at each cycle. In event-driven concurrent systems the same capability should be available, and can be accomplished with execution time clocks and timers.

This need for managing execution time is recognized in standards related to real-time systems. The POSIX standard [4] defines services for execution time measurement and budget overrun detection, and its associated real-time profiles [5] require implementations to support these services. The recent Ada 2005 standard introduced a number of new real-time services intended to provide applications with a higher degree of flexibility. In particular this standard defines capabilities for measuring the execution time of individual tasks, and the ability to detect and handle execution-time budget overruns.

As real-time applications evolve towards an increased complexity level, issues such as composability of independently developed application components and support for legacy code introduce the need for supporting different levels of hierarchy in the scheduling mechanism, leading to a hierarchical concurrency model with different layers, and with capabilities for establishing boundaries for the protection of different parts of the application. In this context of hierarchical scheduling it is often required to bound the execution time of a group of activities that are inside the same protection boundary, so that they cannot interfere with other activities in other protection boundaries by using up more resources than they should. This need introduces a requirement on the underlying implementation to support the measurement of the execution times of groups of tasks, and the handling of potential budget overruns, in a way similar to what is usually done for individual tasks.

Following this general requirement, the Ada 2005 standard defines services for execution-time budgets for groups of tasks, and is now a step forward in relation to the real-time extensions to POSIX, which still has no such service.

In this paper we propose an implementation of a mechanism to support execution-time budgets for thread groups inside a POSIX operating system. The API of this implementation could be used as a basis for a future extension to POSIX. It will also be used to implement the task group budgets defined in Ada 2005. The architecture and details of the implementation are shown, as they may be useful to other implementers of this functionality defined in the new standard. Some performance metrics are provided.

The paper is organized as follows. Section 2 discusses the current services that are available in the platform chosen for this implementation, MaRTE OS and GNAT, and that are related to thread group budgets. Section 3 introduces the services designed to represent sets of threads. Section 4 discusses the implementation of the execution time clocks for groups of threads, while Section 5 does the same for budgets and their associated handlers. Section 6 provides some performance metrics and, finally, Section 7 gives our conclusions.

2 Background

The implementation of execution time budgets for thread groups presented in this paper has been developed in

[1] This work has been funded by the Plan Nacional de I+D+I of the Spanish Government under grant TIC2005-08665-C03 (THREAD project), by Ada Core, and by the European Union's Sixth Framework Programme under contracts FP6/2005/IST/5-034026 (FRESCOR project) and IST-004527 (ARTIST2 NoE). This work reflects only the author's views; the EU is not liable for any use that may be made of the information contained herein.

This paper previously appeared in ACM Ada Letters, Volume XXVII, Number 2, August 2007; reprinted with permission.

MaRTE OS [1] [2], which is a real-time operating system (RTOS) that follows the POSIX.13 [5] minimum real-time system profile, and is mostly written in Ada. It is available for the ix86 architecture as a bare machine, and it can also be configured as a POSIX-thread library for GNU/Linux. The GNAT run-time library has been adapted to run on top of MaRTE OS, which is itself being extended in a joint effort between Ada Core and the University of Cantabria with the objective of providing a platform fully compliant with Ada 2005, available for industrial, research, and teaching environments. The implementation of thread group budgets presented in this paper is part of the effort to achieve this objective.

Two of the new Ada 2005 real-time services are closely related to the thread group budgets and are already available in MaRTE OS and GNAT [3]:

- *Timing events* are defined in Ada 2005 as an effective and efficient mechanism to execute user-defined time-triggered procedures without the need to use a task. They are very efficient because the event handler may be executed directly in the context of the interrupt handler, avoiding the need for a server task.
- Execution time clocks and timers are defined in Ada 2005 as a standardized interface to obtain the execution time consumption of a task, together with a mechanism that allows creating handlers that are triggered when the execution time of a task reaches a given value, providing the means to execute a user-defined action when the execution time assigned to a specific task expires.

Timing events have been implemented in MaRTE OS through a service that we call “timed handlers”, which are not only useful to implement their Ada counterpart, but are also useful to other applications as a general-purpose RTOS mechanism.

MaRTE OS supports the execution-time clocks and timers defined in POSIX.1, which would be appropriate to implement their couterparts in Ada. However, the timers defined in POSIX to detect execution time overruns use an operating system signal to notify about their expiration. Signals are a very scarce resource inside an RTOS. Besides, the signal is usually handled through a thread that is waiting to accept the signal, but this is a mechanism that introduces relatively high overheads, mainly due to the need for the handler to be a thread, with the associated costs in context switches. This leads to the same reason for introducing the new "timing events" mechanism for regular time management.

As a consequence, the Ada implementation of execution time clocks and timers has been achieved in MaRTE through the "timed handler" mechanism, which allows a direct handling of the event inside the hardware timer interrupt handler, thus avoiding the use of a signal and the subsequent double context switch that would be necessary otherwise.

To implement thread group budgets inside MaRTE OS we will follow an approach similar to that followed for execution time budgets for individual threads, creating the appropriate execution time clocks for thread groups and extending the "timed handler" mechanism to also support these new clocks.

3 Thread sets

Before creating the execution time clocks for thread groups or sets, it is necessary to specify a mechanism to represent the groups themselves. Instead of defining a mechanism specific to execution-time clocks, we have chosen to create an independent RTOS object that represents a group of threads. In this way, we will be able to address future extensions that require handling groups of threads using these same objects. Examples of such new services might be related to the requirements for supporting hierarchical scheduling, for instance to suspend or resume a group of threads atomically.

A thread set is implemented by a record that may be extended in the future to add functionality. This record has the following fields:

- *Set* : A list of the threads belonging to the set.
- *Iterator*: A reference to the current thread in the list, used when iterating through marte_threadset_first and marte_threadset_next.

A restriction has been made so that a thread can belong to only one thread set. This restriction is also made in the Ada 2005 standard, and its rationale is that in the hierarchical scheduling environment for which thread groups are useful, threads only belong to one specific scheduling class, and therefore to one specific set. This restriction allows a more efficient implementation, because at each context switch only one of the *Consumed_Time* fields of the set to which the running thread belongs needs to be updated.

Threads can be added/removed to/from a thread set dynamically. Every thread has a pointer in its thread control block (TCB) to the set it belongs to. This field is *null* if the thread doesn’t belong to any thread set.

The C language API to manage thread sets from the application level is the following:

```
// create an empty thread set
int marte_threadset_create (
        marte_threadset_id_t *set_id);
// destroy a thread set
int marte_threadset_destroy (
        marte_threadset_id_t set_id);
// empty an existing thread set
int marte_threadset_empty (
        marte_threadset_id_t set_id);
// add a thread to a set
int marte_threadset_add (
        marte_threadset_id_t set_id,
          pthread_t thread_id);
```

```
// delete a thread from a set
int marte_threadset_del (
        marte_threadset_id_t set_id,
        pthread_t thread_id);
// check thread membership
int marte_threadset_ismember (
        marte_threadset_id_t set_id,
        pthread_t thread_id);
// reset the iterator and get the first thread id
int marte_threadset_first (
        marte_threadset_id_t set_id,
        pthread_t *thread_id);
// advance the iterator and get next thread id
int marte_threadset_next (
        marte_threadset_id_t set_id,
        pthread_t *thread_id);
// check whether the iterator can be advanced
int marte_threadset_hasnext (
        marte_threadset_id_t set_id)
// get the set associated with the given thread
int marte_threadset_getset (
        marte_threadset_id_t *set_id)
        pthread_t thread_id);
```

4 Execution time clocks for thread groups

To implement execution time clocks for groups of threads we add the following information to the object that represents a thread set:

- *Consumed_Time*: CPU-time consumed for all the task in the group. Every time a thread of a given set leaves the CPU, the time consumed by this task since its last activation is added to the *Consumed_Time* of its thread set, even if there is no timed event associated with it, because the value of the execution-time clock may be read at any time by the application.
- *Group_Timed_Event* : A reference to the internal RTOS execution time event, used by the scheduling mechanism. A set can be associated with at most one such event.

The API to obtain an execution-time clock from a thread set is:

```
// destroy a thread set
Int marte_getgroupcpuclockid (
        marte_threadset_id_t set_id,
        clockid_t *clock_id);
```

The returned id represents a clock that can be read and set through the standard POSIX API for clocks, i.e., using functions clock_gettime, clock_settime, ... They can also be used as the base for POSIX timers and MaRTE OS timed events as any other clock defined in the system. They can not however be used as the base for the clock_nanosleep operation, as is also the case with the single-thread CPU-time clocks. POSIX leaves this behavior as unspecified and Ada does not define execution time as a type that can be used in the equivalent delay statements.

POSIX requires type clockid_t to be defined as an arithmetic type, and therefore clock ids are implemented using a unsigned number of 32 bits. The value stored in a clock id can have different interpretations:

- Special values for the regular calendar-time clock CLOCK_REALTIME, the execution time clock of the current thread CLOCK_THREAD_CPUTIME_ID, and the monotonic clock CLOCK_MONOTONIC.
- A pointer to a thread control block when the clock is a thread CPU-time clock of a particular thread.
- A pointer to a thread set when it is a thread group clock.

5 Timed events based on a group clock

Group clocks can be used as the base of timers and timed handlers. When a timer or a timed handler is armed, a MaRTE OS timed event is enqueued in the system event queues. Time-based events in MaRTE OS are of two kinds: standard time and execution time. They are kept in separate priority queues because they cannot be compared with each other for ordering. Events based on group clocks are a special case of execution time events. An execution time event has the following information:

- *CPU_Time*: The event will expire when the execution time consumed by the associated task reaches this value
- *Group_Expiration_Time*: The event will expire when the *Consumed_Time* field of the task set associated with the event reaches this value. This field is only used in events based on a group clock.
- *Is_Based_On_Group_Clock*: This is a boolean used to identify events based on group clocks
- *Base_Clock*: A clock id representing the clock used as the timing base of the event. It could be a thread CPU-time clock or a group clock.
- *Task_Where_Queued* : A pointer to the task that has queued the event.

Execution time events are kept in a queue associated with the task on which the event is based on, and stored as the *CPU_Time_Timed_Event_Queue* in the task control block. Every time a new thread gets the CPU, the events at the head of the standard-time events queue and of the running task´s *CPU_Time_Timed_Event_Queue* queue are compared. The hardware timer is programmed to expire at the most urgent of the two.

Events based on group clocks are special CPU-time events that "jump" between the *CPU_Time_Timed_Event_Queue* of the threads in the group. Each time the system schedules a task included in a thread set that has an event associated, the following actions are performed in the Do_Scheduling internal kernel operation:

```
- - Set CPU_Time of the event according to the
- - time consumed by T
T.Set.Group_TE_Ac.CPU_Time :=
   T.Used_CPU_Time +
   (T.Set.Group_TE_Ac.Group_Expiration_Time -
    T.Set.Consumed_Time);
```

```
 - - Move Group_TE_Ac from one task to another
if T.Set.Group_TE_Ac.Task_Where_Queued /= null
then
  - - Dequeue from the list it was queued
  Dequeue (T.Set.Group_TE_Ac,
      T.Set.Group_TE_Ac.Task_Where_Queued.
      CPU_Time_TEs_Q);
end if;
- - Enqueue in T's list
Enqueue_In_Order (T.Set.Group_TE_Ac,
      T.CPU_Time_TEs_Q);
T.Set.Group_TE_Ac.Task_Where_Queued := T;
```

Dequeue and enqueue operations are very fast, because the number of CPU-time events associated to a task usually will be very small, either one or two: a CPU-time event and a "group event". Consequently the number of extra operations required at each context switch to manage these clocks is kept small, and the implementation can efficiently schedule the threads with an acceptable overhead, as can be seen in the following performance metrics section.

6 Performance metrics

The support for group budgets has already been implemented in MaRTE OS. Execution time accounting introduces a small overhead: enabling this service in MaRTE OS increments the context switch time by less than 5%. Group execution time accounting increments the context switch time by another 4%, representing a total of 9% increment with respect to a system with no CPU-time accounting in an x86 architecture.

The overheads of the budget overrun detection are also relatively small. Table 1 shows a comparison of the overheads of two detection mechanisms, as measured in a 3.4GHz Pentium IV. The first one is implemented using a regular POSIX timer that sends a signal when the budget expires, and a handler thread that blocks waiting to accept the signal. The second mechanism is implemented using the new timed handler service. We can see that the overhead of the second mechanism is much smaller.

Table 1 Overhead of budget overrun notification mechanism

Metric	Time (⌠s) (using timer and auxiliary thread)	Time (⌠s) (using timed handlers)
From user´s thread to handler	1.1	0.4
From handler to user´s thread	0.8	0.7
Total time:	1.9	1.1

7 Conclusion

As the complexity of real-time systems evolves, hierarchical scheduling and partitioning are mechanisms used to cope with it, by helping in establishing protection boundaries and easing the composability of independently-developed application components. One of the requirements of this partitioning is the time protection among the different groups of tasks in the hierarchy, which can be achieved by using thread group budgets as those specified in the new Ada 2005 standard.

This paper has presented an implementation of the support needed to provide such budgeting services in a real-time operating system called MaRTE OS. The paper describes the architecture and details of the implementation, together with the rationale for the main design decisions, so that this information can be used by other implementers of this functionality, either as part of Ada run-time systems, or as part of a general-purpose RTOS. The implementation has proven to be straightforward, and the overheads introduced are small, both in the context switch times and in the budget overrun notification mechanism.

As future work, the functionality defined in Ada 2005 for group budgets will be implemented. It is anticipated that support for the Ada group budgets will be a simple package built on top of the MaRTE OS implementation described in this paper.

References

[1] Aldea Rivas M. and González Harbour M. *MaRTE OS: Minimal Real-Time Operating System for Embedded Applications*. Universidad de Cantabria. http://marte.unican.es/

[2] Aldea Rivas M. and González Harbour M. *MaRTE OS: An Ada Kernel for Real-Time Embedded Applications.* Proceedings of the International Conference on Reliable Software Technologies, Ada-Europe-2001, Leuven, Belgium, Lecture Notes in Computer Science, LNCS 2043, May, 2001, ISBN:3-540-42123-8, pp. 305,316.

[3] Aldea Rivas M. and Ruiz J.F.. *Implementation of new Ada 2005 real-time services in MaRTE OS and GNAT*. International Conference on Reliable Software Technologies, Ada-Europe-2007, Switzerland.

[4] IEEE Std. 1003.1:2004 Edition, *Information Technology — Portable Operating System Interface (POSIX)*. The Institute of Electrical and Electronics Engineers.

[5] IEEE Std. 1003.13-2003. *Information Technology – Standardized Application Environment Profile- POSIX Realtime and Embedded Application Support (AEP)*. The Institute of Electrical and Electronics Engineers.

[6] S. Tucker Taft, Robert A. Duff, Randall L. Brukardt, Erhard Ploedereder, Pascal Leroy (Eds.). *Ada-2005 Reference Manual. Language and Standard Libraries.* International Standard ISO/IEC 8652/1995(E) with Technical corrigendum 1 and Amendment 1. Springer, Number 4348 in Lecture Notes in Computer Science, Springer-Verlag (2006).

Session: Beyond Ada 2005

Chair: Jorge Real

Rapporteur: Stephen Michell

1 Session Goals :

To consider future directions in computing, and what changes would be required for Ada to effectively use new features.

Related Papers

1. *Beyond Ada2005: Allocating Tasks to Processors in SMP Systems*; A.J.Wellings and A. Burns.
2. *Suggestions for Stream Based Parallel Systems in Ada*; M. Ward and N.C. Audsley

2 Stream-Based Parallelism

Neil Audsley gave a look at a possible future in computation based on massive parallel architectures. The presentation began with current architectures, including

a) single CPU with L1, L2 cache and memory, and

b) double CPU with L1, L2 cache and memory, cache coherence at L2 memories

The presentation noted that these architectures are unscalable beyond a few (4-6) CPUs, because the replication of processors (CPU) on each chip, separation of L2 cache onto dedicated chips, distances and switching speeds of circuits when using multiple chips increase delays and power requirements.

An alternative view was presented that was called “System on a chip". Such a system has:

- Heterogeneous CPUs,
- Non uniform memory,
- Special devices.

It is expected that this will soon be followed by "Network on a chip", which consists of:

- Multiple systems-on-chip connected by networks
- No common notion of time
- Packet switched network

An example of such a system has been developed by the authors, that amounts to "Field Programmable System on a chip". Such a system is highly reprogrammable and can be reprogrammed in milliseconds, using an almost Ravenscar compliant system.

The authors identified some issues for the Ravenscar Tasking Profile.

Open Issue 1.1: Lack of a shared lock for Protected Entries

This was expressed as Ravenscar's restriction to a single entry per protected object or a single caller task per entry, but discussion highlighted that the problem is fundamental in Ada's specification of protected operations.

Protected functions in Ada permit a shared access to a protected object, but lack any synchronization. Protected entries provide synchronization, but lack the ability for a collective release of waiting tasks and each released task maintains a sequential lock. The need in highly parallel systems is to release collections of tasks that will read their dedicated data and not update protected data, hence behaving as a function once released.

The Ravenscar restrictions of a single entry and a single queue element per entry exacerbate this problem. It was agreed that this was a problem that requires a proposal to the Ada Rapporteur Group to solve these issues. Solutions could resemble a *pragma Simultaneous_Release*, or the addition of functions that block to protected objects.

The workshop agreed that this deserved further study.

2 Synchronous Multiprocessing

Andy Wellings presented a summary of the paper “Beyond Ada2005: Allocating Tasks to Processors in SMP Systems” and then lead a discussion on the topic through an interactive slide presentation. The paper, presentation and discussion assume a model of a shared memory multiprocessor environment and additions required for Ada 2005 to better support such an environment.

The author noted that Ada nominally addresses the multiprocessor environment, but assumes that there is an OS-level or implementation-level of support that simplifies the view of multiprocessing to make it seamless. Specifically, the paper notes that Ada is currently silent on how the runtime maps tasks to specific processors, and proposes the use of pragmas to let an application guide such mappings.

The authors claim that better schedulability can be obtained by supporting static allocation of tasks to CPUs. They also claim that the approach is not scalable to multicore architectures that are Non Uniform Memory Access (NUMA). The authors also note that there is still no standardization of support for SMP in OS community, which affects any choices that Ada makes because Ada implementations may rely upon services that are not

This paper previously appeared in ACM Ada Letters, Volume XXVII, Number 2, August 2007; reprinted with permission.

supplied, or may make choices that differ significantly from those eventually chosen by an OS. The challenge is to provide set of mechanisms that can be both expressive enough to support a wide range of application requirements, yet be implemented on a wide range of OS's.

Platform variability is a very significant issue for multiprocessor systems. An assumption is made that a concurrent program running on a SMP system will often not be the only program executing, that the hardware resources available to it will not be constant throughout the execution of a single execution, and that some processors may have capabilities or interfaces that are not available to other processors.

For example,

a) An underlying operating system may dynamically change set of processors allocated to a program during execution and may or may not inform the executing program of such changes.

b) There may be hardware registers, interfaces to the external environment or interrupts available to some processors but not to all.

It is hoped that such changes would be done in a safe manner, but at present there is no language mechanism to manage these issues. The workshop decided that the minimal level of support that a program requires is to be able to determine how many processors are available to it. A proposed Ada service is shown in paper [1].

Another issue raised was that Ada 2005's support of task groups should interoperate with processor affinity. An extension of *Set_Affinity* to a task group would be useful. Another issue raised was that some aspects of memory maps may be processor-specific, and that ways to specify memory affinities should be considered. There were no specific set of calls proposed to provide such capability.

Throughout the presentation and discussion, there were a number of "Open Issues" that were raised and discussed.

Open Issue 2.1: Should the mapping of tasks be by-partition?

There was general agreement that this was the desired model.

Open Issue 2.2: Should there be Affinity Inheritance?

There was some discussion but no strong conclusion. It was generally agreed that such a model would work, in that nested tasks would start with the same processor affinity and could explicitly change that affinity with a call. It was noted that a pragma, such as *pragma priority* could be used for static affinity control.

Open Issue 2.3: Dispatching policies

There was agreement that dispatching policies must be partition-wide. A discussion was held about specific Ada dispatching policies and how they would be affected by the SMP model.

a) Dispatching policy *FIFO_Within_Priorities* should imply that a task can be migrated between its allocated processors whenever it is preempted.

b) Dispatching policy *Non_Preemptive_FIFO_Within_Priorities* should mean that a task, once dispatched to a processor, will not be migrated from that processor while it is still executable (because it cannot be preempted).

c) The meaning of the dispatching policy *EDF_Across_Priorities* is unclear if the tasks assigned to the priority range have a disjoint set of processors.

d) This raises the need for a new dispatching policy, *FIFO_Within_Priorities_Without_Migration*, where a preempted task cannot be migrated from the processor from which it was preempted while it is still runnable.

The discussion also considered the ramifications of affinity to scheduling policies. The ARM view of priorities states that high priority tasks ready for execution should always be executed in preference to lower priority tasks. Examples were given where a high priority task executing on a single processor (say HI with affinity {A}) could preempt a medium priority task (say MED on A with affinity {A, B}).

Open Issue 2.4: Interrupt handlers, Protected Objects & Tasks.

Ada's nominal mapping of interrupts is to protected objects, but tasks also often initiate and complete interrupt-level operations. If interrupts are processor-specific, there must be a way to map protected objects and tasks to the processor. An alternative procedure *Set_Handler* was proposed that would include the affinity mapping, but it was noted that task-processor affinity could also be a requirement. A further complication would result if a single task called 2 protected objects that had different affinities.

The workshop decided that this was an area of interest and for further study.

Open Issue 2.5: Consistent notion of time.

Timers and relative delay were discussed and considered to be consistent. Absolute notions of time could be a problem, but should be satisfactory within a single partition. CPU time, however, could be problematic as processors may not all have the same clock speeds, and reduction or increase on the processor set could hinder calculations that optimize CPU-time.

It was agreed that at a minimum should be standardized for symmetric multiprocessing with static processor allocations.

Open Issue 2.6: Is it important how an OS manages SMP's?

The consensus was that Ada programs sit above OS implementations and cannot rely upon specifics of the OS-to-processor decisions.

Open Issue 2.7: Mapping Tasks to Processors:

The next discussion considered the mapping of tasks to processors. There was a general consensus that the mapping should be task-based, as opposed to partition-based. There was also sentiment that such a mapping should include mapping of data-specific regions, cache description and interrupts to processors. The following mapping choices for tasks-processors were enumerated,

1) Task → Processor
2) Task → {Processor}
3) {Task} → Processor
4) {Task} → {Processor}
5) {Task} → {Processor} + return to same processor.

It was agreed that the mapping proposal enumerated above is a reasonable beginning, but that pragmas should be included for the static mappings and memory mappings should be considered.

The workshop noted that affinity and pre-emption can lead to cascading preemptions. A case in point,

- HI on A, MED with affinity {A, B}, LOW on any.
- HI preempts, but can't preempt LOW because LOW is on a processor for which HI has no affinity,
- HI therefore preempts MED which must then preempt LOW.

Other scenarios can be constructed where priority inversion occurs, i.e. HI preempts MED but MED cannot preempt LOW because MED has no affinity for the processor executing LOW.

Round-robin scheduling was discussed, and it was concluded that as long as all tasks participating in the round-robin at the same priority level had identical processor affinities, placing a task that has just finished its quantum at the end of the queue for all processors in the affinity set would suffice.

EDF was thought to be generally ok, but will cause preemption cascading. Further research is required.

Open Issue 2.8: What happens if OS removes a processor?

This is a serious issue if the processor causes significant perturbations in the affinity set of some tasks, such as giving a task a null affinity set. The call-back notification discussed earlier may suffice, as long as there was prior notification of the removal so that tasks could synchronously change their affinity sets.

Open Issue 2.9: Asynchronous Task Control and Affinity

There was a proposal to be able to add to *Aynchronous_Task_Control* the ability to change affinity. This proposal received insufficient support.

Open Issue 2.10: Protected Objects and Processor Affinities.

There are some significant issues in giving protected objects affinities. The requirement is clear since processor-specific mappings such as interrupts and registers may be utilized with no task thread, or may be called by a task without affinity for the processor in question. This is most likely if the implementation had proxy execution of protected entries, and a task with the wrong processor affinity tried to execute a protected entry on behalf of another task. There is, however, no current concept of *Protected_Object_ID* similar to *Task_ID* to build such a mapping.

It was noted that the existing Ada pragma *Attach_Handler* requires extension to include processor information where applicable. Similarly, a pragma to provide affinity could provide static affinities for protected objects.

It was decided that this topic needed further research.

3 Conclusions

As the session wrapped up, it was decided to continue developing proposals for the next workshop, and for the Ada Rapporteur Group to consider as they are developed.

Beyond Ada 2005: Allocating Tasks to Processors in SMP Systems

A.J. Wellings and A. Burns

Department of Computer Science, University of York, UK; email: {andy,burns}@cs.york.ac.uk

Abstract

Ada 2005 has added no new facilities to support applications that want to run on multiprocessor systems. Following the example set by Ada 95, the language facilitates multi-processor implementations but provides no direct support for an application-controlled mapping of tasks to processors. Such partitioning is often required to obtain feasible real-time systems. This paper argues that multiprocessors systems are becoming so prevalent that the current position is no longer tenable. A proposal for minimal support is presented.

1 Introduction

Multiprocessors system are becoming more prevalent. In particular SMP systems are often the default platform for large real-time systems rather than a single processor system. The scheduling of processes on these systems can be

1. global – all processors can execute all processes
2. fully partitioned – each process is executed only by a single processor; the set of processes is partitioned between the set of processors
3. mixed – each process can be executed by a subset of the processors; hence the tasks set may be partitioned into groups and each group can be executed on a subset of the processors.

The Ada Reference Manual allows a program's implementation to be on a multiprocessor system. However, it provides no direct support that allows programmers to partition their tasks onto the processor in the given system. The following ARM quotes illustrate the approach.

> "NOTES 1 Concurrent task execution may be implemented on multicomputers, multiprocessors, or with interleaved execution on a single physical processor. On the other hand, whenever an implementation can determine that the required semantic effects can be achieved when parts of the execution of a given task are performed by different physical processors acting in parallel, it may choose to perform them in this way." ARM Section 9 par 11.

This simply allows multiprocessor execution and also allows parallel execution of a single task if it can be achieved, in effect, "as if executed sequentially".

> "In a multiprocessor system, a task can be on the ready queues of more than one processor. At the extreme, if several processors share the same set of ready tasks, the contents of their ready queues is identical, and so they can be viewed as sharing one ready queue, and can be implemented that way. Thus, the dispatching model covers multi-processors where dispatching is implemented using a single ready queue, as well as those with separate dispatching domains." D.2.1 par 15.

This allows the full range of partitioning identified above. However, currently the only way that an implementation can provide the mechanisms to allow the programmers to partition their tasks amongst the available processors is via implementation-defined pragmas, or non standard library packages.

This paper argues that multiprocessor systems are becoming so prevalent that it is now time for the language to provide more direct support.

Unfortunately, as of yet, there has been no standardisation of support for multiprocessor systems in the operating system community. Hence, if Ada is being implemented on top of a real-time operating system, it is difficult to know what facilities it can rely on. Consequently, the challenge is to provide a set of mechanisms that can be both expressive enough to support a wide range of application requirements and yet can be implemented (possibly with degraded services) on a wide range of operating systems.

This paper proposes the introduction of a new package System.Processor_Elements to capture the interface between the programmer and the underlying system's multiprocessor architecture. A new pragma, Affinity is also introduced. The focus is on support for SMP (Symmetric MultiProcessor) Systems.

The paper is structured as follows. In Section 2 we present the main motivations for wanting to provide more explicit support for multiprocessor systems. In Section 3 we briefly review the support that has been discussed or provided by current operating systems. Drawing on this work, we then present (in Section 4) some *initial* thoughts on how to

This paper previously appeared in ACM Ada Letters, Volume XXVII, Number 2, August 2007; reprinted with permission.

integrate multiprocessor support into Ada. Finally we present our conclusions.

2 Motivation

Whilst many applications do not need more control over the mapping of tasks to processors in an SMP environment, there are occasions when such control is important. They include:

- To allow more flexible approaches to scheduling. – Although the state of the art in schedulability analysis for multiprocessor systems continues to advance [2], the current state is such that partitioned systems offer more guaranteed schedulability than global systems. Quoting from [3]:

  "The choice between global and partitioned approaches to multiprocessor scheduling is a conundrum. Setting aside pragmatic questions about queue contention overhead and differences in cache behavior, the theoretical results are equivocal.

  In favor of global scheduling, it has long been known from queueing theory that single-queue (global) FIFO multiprocessor scheduling is superior to queue-per-processor (partitioned) FIFO scheduling, with respect to average response time.

  Apparently in favor of partitioned scheduling, the application of well-known single processor scheduling algorithms appears superior to the global application of those same algorithms for some task sets with hard-deadlines.

  For example, it is known that all periodic implicit-deadline task sets with utilization below $m(2^{1/2} - 1)$ can be scheduled on m processors using a first-fit-decreasing-utilization (FFDU) partitioning algorithm and and local rate monotonic scheduling, but Dhall's example shows that there are hard-deadline periodic task sets with total utilization arbitrarily close to 1.0 that cannot meet all deadlines if scheduled on m processors using global rate monotonic scheduling.

  Dhall's example also applies to global EDF scheduling, yet FFDU partitioned EDF scheduling is guaranteed up to utilization $(m + 1)/2$. However, the supposed advantage of partitioned scheduling above disappears if one considers hybrid global priority schemes. The Dhall example can easily be handled by the *EDF − U S(1/2)* or *EDF($k_m$in)* schemes, in which top priority is given to a few "heavy" tasks, as can any implicit deadline sporadic task system with utilization up to $(m+1)/2$. This is exactly the same bound as for FFDU partitioned scheduling!

  The experiments we performed on large numbers of pseudo-randomly generated task sets were intended to provide some additional evidence on which to base a choice between these two approaches. From those experiments, statistically, the chance of being able to satisfy all the deadlines of a randomly chosen periodic or sporadic task set appears to be highest with partitioned scheduling. In particular, the partitioned EDF scheduling appeared to be the overall best performer in this statistical sense. At the same time, there are certainly specific task sets where global scheduling is more effective.

  While the schedulability tests used in the experiments probably could be improved, it remains unclear whether they can be improved enough to erase the statistical margin of partitioned scheduling with the available schedulability tests."

- To support temporal isolation. – Where an application consists of tasks of mixed criticality level, some form of protection between the different levels is required. The strict typing model of Ada provides a strong degree of protection in the spatial domain. The CPU budgeting facility provides a limited form of temporal protection but at the expense of flexibility. More flexible temporal protection is obtainable by allowing tasks in each criticality level to be executed on partitions of the processor set.

- To obtain performance benefits. – For example, dedicating one CPU to a particular process will ensure maximum execution speed for that process. Restricting a process to run on a single CPU also prevents the performance cost caused by the cache invalidation that occurs when a process ceases to execute on one CPU and then recommences execution on a different CPU [4].

- To be able to respond to dynamic changes to the processor set. – In a parallel computing environment the set of processors allocated to an application may vary depending on the global state of the system. An application may be able to optimize its algorithms if it is informed when these changes in the processor set occur.

One of the application areas where use of Ada remains strong is in high-integrity systems. It is important to anticipate how the requirements on these systems will change over the coming years so that we can ensure that Ada remains competitive.

Currently there is limited use of general multiprocessor shared memory systems in Safety Critical Systems. Traditionally, where multiprocessors are required they are used in a distributed processing mode: with boards or boxes interconnected by communications busses, and bandwidth allocation, and the timing of message transfers etc carefully managed. This "hard" partitioning simplified certification and testing since one application cannot affect another except through well-defined interfaces. More recently, there has been a move towards more integrated distributed systems where functions are more distributed across a single computing infrastructure (e.g. Integrated Modular Avionics). The goal of this approach is to save space and weight, reduce wiring, provide cheaper fault toleranc and reduce overall costs. Partitioning here is "softer" and is supported by a combination of hardware and software techniques (e.g. memory management support to protect

address spaces, some form of CPU budgeting to enforce temporal firewalls, and TDMA on the network).

There has been some use of shared memory modules between processors but access to these memory modules are very restricted and typically only used to coordinate computational activity. Where it has been necessary to use an SMP, only one processor has been enabled[1].

However, there is evidence that future systems will use SMP. For example, the LynxSecure Separation Kernel has recently been announced. The following is taken verbatim from their web site[1]:

- Optimal security and safety – the only operating system to support CC EAL-7 and DO-178B level A
- Real time – time-space partitioned real-time operating system for superior determinism and performance
- Virtualization technology – supports multiple heterogeneous operating system environments on the same physical hardware
- Highly scalable – supports Symmetric MultiProcessing (SMP) and 64-bit addressing for high-end scalability
- Support for open standards – supports 100% binary compatibility for Linux or POSIX-based software application to migrate to a highly robust, secure environment
- Faster time to market – enables developers to begin early development for secure applications

This work has been undertaken by Intel and LynuxWorks to demonstrate the MILS (Multiple Independent Levels of Security/Safety) architecture[2].

3 Review

Although POSIX currently does not provide specific support for SMP systems, the issue has been raised [5]. POSIX.1 defines the ”Scheduling Allocation Domain” as the set of processors on which an individual thread can be scheduled at any given time. POSIX states that [6]:

- “For application threads with scheduling allocation domains of size equal to one, the scheduling rules defined for SCHED FIFO and SCHED RR shall be used;”
- “For application threads with scheduling allocation domains of size greater than one, the rules defined for SCHED FIFO, SCHED RR, and SCHED SPORADIC shall be used in an implementation-defined manner.”
- “The choice of scheduling allocation domain size and the level of application control over scheduling allocation domains is implementation-defined. Conforming implementations may change the size of scheduling allocation domains and the binding of threads to scheduling allocation domains at any time.”

With this approach, it is only possible to write strictly conforming applications with real-time scheduling requirements for single-processor systems. If an SMP platform is used, there is no portable way to specify a partitioning between threads and processors.

Additional APIs have been proposed but currently these have not been standardized. The approach has been to set the initial allocation domain of a thread as part of its thread-creation attributes. The proposal is only draft and so no decision has been taken on whether to support dynamically changing the allocation domain.

Since Kernel version 2.5.8, Linux has provided support for SMP systems [4] via the notion of CPU affinity. Each process in the system can have its CPU affinity set according to a CPU affinity mask. A process’s CPU affinity mask determines the set of CPUs on which it is eligible to run.

```
#include <sched.h>

int sched_setaffinity(pid_t pid,
    unsigned int cpusetsize,
    cpu_set_t *mask);

int sched_getaffinity(pid_t pid,
    unsigned int cpusetsize,
    cpu_set_t *mask);

void CPU_CLR(int cpu, cpu_set_t *set);

int CPU_ISSET(int cpu, cpu_set_t *set);

void CPU_SET(int cpu, cpu_set_t *set);

void CPU_ZERO(cpu_set_t *set);
```

A CPU affinity mask is represented by the cpu_set_t structure, a “CPU set”, pointed to by the mask. Four macros are provided to manipulate CPU sets. CPU_ZERO clears a set. CPU_SET and CPU_CLR respectively add and remove a given CPU from a set. CPU_ISSET tests to see if a CPU is part of the set. The first available CPU on the system corresponds to a cpu value of 0, the next CPU corresponds to a cpu value of 1, and so on. A constant CPU_SETSIZE (1024) specifies a value one greater than the maximum CPU number that can be stored in a CPU set.

sched_setaffinity sets the CPU affinity mask of the process whose ID is pid to the value specified by mask.

If the process specified by pid is not currently running on one of the CPUs specified in mask, then that process is migrated to one of the CPUs specified in mask.

sched_getaffinity allows the current mask to be obtained.

An error is returned if the affinity bitmask mask contains no processors that are physically on the system, or

[1] http://www.lynuxworks.com/rtos/secure-rtos-kernel.php.

[2] See http://www.intel.com/technology/itj/2006/v10i3/5-communications/6-safety-critical.htm.

cpusetsize is smaller than the size of the affinity mask used by the kernel.

The affinity mask is actually a per-thread attribute that can be adjusted independently for each of the threads in a thread group. The value returned from a call to gettid (get thread id) can be passed in the argument pid.

Other operating systems provide slightly different facilities. For example IBM's AIX allows a kernel thread to be bound to a particular processor[3]. Further more, the set of processors (and the amount of memory) allocated to a partition in AIX can change dynamically. In AIX a partition appears to be a subset of resources allocated to a particular subsystem.

The Expert Group responsible of development of the Real-Time Specification for Java (JSR 282) is also considering the appropriate level to support SMP systems. The proposal given here is compatible with their current view.

4 Proposal

In the general case, the following may be supported by the underlying platform (operating system and hardware).

1. An application program may be allocated (by the operating system) the full set of the processors in the system or only a subset of them. An initial allocation is performed at the start of program execution time.
2. The operating system may only support global scheduling of threads or it may allow threads to be constrained to one or more processors in the set allocated to the program.
3. The operating systems may dynamically change the allocation of processors allocated to a program during the program's execution. If it does this, it is done in a safe manner.
4. Mechanisms may be provided by the operating system to inform the application (if the operating system supports task to processor allocation) or they may not (if it only supports global scheduling).

From an Ada perspective, there are two possible approaches to supporting task to processor allocation:

1. associate Ada partitions with processor sets
2. associate individual tasks with processor sets

We use the latter approach, as partitions in Ada are more a unit of distribution (or at least implies that each partition executes in a separate address space) and are not first class entities. Here, we are concerned with entities that share memory. Hence the mechanisms we define here are on a per partition basis and we allow tasks to set their processor affinity.

[3] See http://publib.boulder.ibm.com/infocenter/pseries/v5r3/topic/-com.ibm.aix.basetechref/doc/basetrf1/bindprocessor.htm

The mechanisms supported by the proposed package (see Figure 1) have been designed with the constraints that should degrade if the program is executing

1. on a single processor system
2. under an operating system which imposes a global partitioning approach.
3. under an operating system that does not change the processor set allocated to a program.

The minimum functionality is for the operating system to allow an Ada program to determine how many processors are available to it.

The API allows for systems that support the dynamic addition and removal of processors from the set allocated to the program. If an operating system does not support this facility then the set will not dynamically change. An operating system is also allowed to maintain a set of logical processors allocated to the program and to transparently change its logical to physical mapping. Again, from the Ada programs perspective the set has not changed. However, it should be noted that this may have an impact on the application if a) it is handling interrupts directly on the processor or b) if the change undermines any feasibility analysis assumptions. For many Ada applications this may not be a problem. In all of the above circumstances Dynamic_Set_Changes_ Supported is set to False in the following package.

If the operating system does support dynamic changes to the processor set, the assumption is that it will inform the Ada program of the changes (e.g. via a signal). The Ada run-time system will pass this information to the application via the calling of a protected procedure. In this circumstances, Dynamic_Set_Changes_Supported is set to True.

The assumption is that the application will maintain its own list of which tasks are mapped to which processors (logical or physical). It will then undertake whatever reconfiguration it deems appropriate.

If a processor fails and the platform cannot transparently recover, the Ada program abnormally ends (with assumed fail stop semantics). Any recovery must be performed outside of the Ada program. This is because a processor failure can leave the application in an inconsistent state (e.g. with a corrupted heap) from which it is unlikely to be able to recover.

The API supports the setting of the affinity tasks by the programmer. If the operating system doesn't support this facility then all of the associated operations, raise the Unsupported_Operation exception, and Processor_ Affinity_ Supported is set to False.

The full API is shown in the AFigure 1, annotated with the semantics of the subprograms. For convenience, the affinity mask is shown as a boolean array. In practice, a more efficient representation of the affinity mask would be needed.

```
with Ada.Task_Identification; use Ada.Task_Identification;
package System.Processor_Elements is
   Affinity_Error : exception;
   Unsupported_Operation : exception;

   type Processors is range 0 .. <<implementation-defined>>;
   - - The number of processors available on this system.
   - - Each processor has a logical Id in the range.
   - - On a single processor system, the range is 0..0

   type Processor_Set is array(Processors) of Boolean;
   - - A set of processors. A boolean set to True, indicates
   - - that the logical processor is included in the set

   function Available_Processors return Processor_Set;
   - - Indicates which of the processors in the system are
   - - current available to the program. In some
   - - systems this will never change, others it may.

   Dynamic_Set_Changes_Supported : constant Boolean := <<implementation-defined>>;
   - - Indicates if the current system might dynamically change the
   - - Available_Processor set

   Processor_Affinity_Supported : constant Boolean := <<implementation-defined>>;
   - - Indicates whether the system allows a task's affinity to be
   - - set by the programmer

   function Set_Default_Affinity(Processors: Processor_Set) return Processor_Set;
   - - Raises Unsupported_Operation if Processor_Affinity_Supported = False
   - - Raise Affinity_Error if Processors is incompatible with Available_Processors

   function Get_Default_Affinity return Processor_Set;
   - - The default affinity is the set of processors that can
   - - execute a newly created task. The initial system default is
   - - the set returned from Available_Processors, i.e. global
   - - scheduling on any of the processors available to the system.
   - - If Processor_Affinity_Supported = False, then this always
   - - returns Available_Processors

   function Set_Affinity(Processors : Processor_Set; TID :Task_Id := Current_Task)
   return Processor_Set;
   - - Sets the affinity for a particular task.
   - - Raises Unsupported_Operation if Processor_Affinity_Supported = False
   - - Raises Affinity_Error if Processors is in conflict with Available_Processors
   - - The new affinity is set immediately if the task is not executable.
   - - If it is current executable,
   - - the new affinity is set when the task next becomes non-executable
   - - Returns the old set allocated???

   function Get_Affinity(TID :Task_Id := Current_Task) return Processor_Set;
   - - Returns the current affinity of the task

   type Change_Handler is access protected procedure(Processor : Processor_Set);

   procedure Set_Available_Processor_Changed_Handler(
   New_Handler : in Change_Handler; Old_handler : out Change_Handler);
   - - Raises Unsupported_Operation if Dynamic_Set_Changes_Supported = False
   - - If the system allows processors to be added to or subtracted
   - - from the Available_Processors, then the program can request
   - - notification of these changes via a call to a protected
   - - procedure. Here a new call of Set_Available_Processor_Changed_Handler
   - - overwrites any previous call. Whenever a change occurs, the
   - - system calls the last set handler.
end System_Processor_Elements;
```

Figure 1 Proposed API

Open Issues

- Defaults – The current proposal has default affinity arrays. In Ada, the default priority of a task is the same as its parent, and a pragma is defined to allow the priority to be set at task creation time. Hence, a pragma such as pragma Set_Affinity(Mask'Access) could be provided.
- Dispatching Policies – Where a task can be executed on more than one processor it may be appropriate to define a new dispatching policy to obtain efficient use of caching. For example, the current policies could be extended and a new one added as follows:
  - FIFO_Within_Priorities. With this policy, a task can be migrated between its allocated processors whenever it is preempted.
  - Non_Preemptive_FIFO_Within_Priorities.With this policy, a task once dispatched to a processor will not be migrated from that processor whilst it is still executable. Furthermore, it cannot be preempted.
  - FIFO_Within_Priorities_Without_Migration. A new policy, a preempted task cannot be migrated from the processor from which it was preempted whilst it is still runnable.
  - EDF_Across_Priorities. It is not clear what this policy means if the tasks assigned to the priority range can be executed on a possible disjoint set of processors.
- Interrupt handling – Some SMPs allow the affinity of an IRQ to be set. Hence, certain interrupt handlers can only run on that processor set (e.g. on Red-hat linux /proc/irq/IRQ#/smp affinity specifies which target CPUs are permitted for a given IRQ (Interrupt ReQuest line) source). An alternative version of the Attach_Handler pragma could be provided to allow the mask to be set. Also a new subprogram in Ada.Interrupts could allow the mask to be set in the dynamic case.
- Asynchronous task control – The current definition of Ada.Asynchronous_Task_Control seems to work adequately for the multiprocessor case. However, setting the affinity of a task to be "no processors" also needs to be considered in this context. In particularly when it is waiting at an accept/select statement.
- Current Processor – A mechanism may be needed for a task to determine the actual processor upon which it is currently executing. Such a facility could be provided in the above package.

5 Conclusions

Historically, Ada has always taken a neutral position on multiprocessor implementations. On the one hand, it tries to define its semantics so that they are valid on a multiprocessor. On the other hand, it provides no direct support for allowing a task set to be partitioned. This paper has argued that multiprocessors are becoming more ubiquitous, and that there are advantages to be gained by allowing the program more control over which task executes where. Unfortunately the POSIX standards do not currently address this issue, and consequently it is difficult to know what mechanisms Ada can rely on existing in the underlying execution platform. Consequently, the paper has proposed an API which can gracefully degrade according to the facilities provided.

Acknowledgements

The authors gratefully acknowledge the contributions of the JSR 282 and JSR 302 Expert Groups where many of the ideas presented in this paper have been discussed (albeit from within a Java context).

References

[1] B.S. Anderson. Saftey critical systems and SMPs, private communication, 2006.

[2] T.P. Baker. An analysis of fixed-priority schedulability on a multiprocessor. *Real-Time Systems*, 32 (1-2):41 – 71, 2006.

[3] T.P. Baker. Global versus partitioned scheduling in multiprocessor systems, private communication, 2006.

[4] Linux Manual Page. sched setaffinity(), 2006.

[5] Michael Gonzalez Harbour. Supporting SMPs in POSIX, private communication, 2006.

[6] Open Group/IEEE. The open group base specifications issue 6, ieee std 1003.1, 2004 edition. IEEE/1003.1 2004 Edition, The Open Group, 2004.

Suggestions for Stream Based Parallel Systems in Ada

M. Ward* and N. C. Audsley

Real Time Systems Group, University of York, York, England; email: (mward,neil)@cs.york.ac.uk

Abstract

Ada provides good support for the implementation of dependable, real-time, control systems. However, its support for other styles of systems is not as good. This paper explores the support available for implementing parallel, stream based systems. The paper presents an implementation of an image manipulation system which highlights deficiencies in the support for such systems in the Ada language. Two additional semantics are proposed for addition to the Ada language which will provide for the needs of these systems. The broadcast semantic allows the same data to be written to several POs simultaneously. The guarded protected function semantic permits several readers to wait on an entry and simultaneously read data from the PO.

1 Motivation

The Ada language [1] has found a niche in the implementation of dependable real-time systems. This has traditionally revolved around the use of periodic tasks to implement control systems. However, there are other styles of system that would benefit from the various attributes that Ada offers. The main advantages of Ada are its support for real-time systems, and its support for concurrency within its semantics. This paper looks at the effects of trying to implement dependable, stream based, parallel systems in the Ada language. These have high performance demands, and require substantial support for efficient implementation.

This paper is presented in two parts: the experience of using Ada to implement such a system; and suggestions to improve the language support for them. Section 2 presents an overview of the implementation of an image manipulation system. This covers the motivation for moving away from the traditional implementation method, through the ideal solution, and the changes required to allow it to be programmed in Ada, to a description of the final system. Section 3 contains suggestions for additions to the Ada language to provide for the needs of such systems that are not supported within the language.

2 The Image Manipulation System

Image manipulation systems work on video streams in real-time. With both source and output streams, the system applies a transformation to the streams. These can be simple, pixel-based, manipulations (e.g. greyscale, sharpen, edge detect), or a more complex, frame-based, transformations (e.g. image warping or morphing). Whilst frame based manipulations need more buffering than pixel based ones, their implementation is similar.

The traditional approach to the implementation of these systems uses graphical libraries implemented on a processor. The processor can be either a general purpose processor or a graphics specific processor which provides support for common graphical functionality. In the general case, these techniques provide ample performance for most image processing techniques. However, the use of these techniques for dependable systems raises a number of issues.

A dependable system needs to be proved to be correct to its specification. This requires analysis of both the functionality of the system, as well as its timeliness (amongst other things). The traditional implementation techniques fall foul of these needs:

- General purpose processors have good best case performance, but due to the architectural features to do this, have poor worst case performance. This limits the available processing power and restricts the overall system performance.
- Specific processors generally do not have the same amount of evidence to prove they are correct that a general purpose processor will have. Though due to the specific instructions they have less need of architectural speed up features.
- Graphics libraries are not written for dependability, generally they are written for speed. This makes proving them correct difficult, especially when their size is considered.

As such, high-performance dependable image manipulation systems are difficult to build. This makes high resolution, high frame-rate image manipulations difficult to do with traditional implementation techniques. This produces a need for a different implementation technique.

2.1 The Problem

The problem arose from a request by BAE Systems to implement a dependable image manipulation system capable of dealing with high-resolution, high-refresh video streams. The system had to be dependable, preferably using

* This work was supported by BAE SYSTEMS

This paper previously appeared in ACM Ada Letters, Volume XXVII, Number 2, August 2007; reprinted with permission.

Ada as limited by the SPARK [2] and Ravenscar [3] subsets. The use of Ada will allow reuse of existing software code. In addition to this, the system needed minimal delay on the output stream. The initial demonstration of this system should be an image warping application (e.g. correcting imperfect optics or pre-distortion for display on shaped surfaces).

2.2 Solution Suggestion

A block diagram of the initial suggested solution can be found in figure 1. This solution relies on parallelism to provide the processing power to perform the transformations. As the manipulation is to be done on a frame by frame basis, the video stream is first read into a screen buffer. To provide for multiple accesses this buffer is replicated a number of times dependant on the needs of the application. This can be a replication of the entire buffer, or buffers that each contain part of the image. The image processing is undertaken by a number of parallel tasks. Each task is responsible for part of the output image, and can access any of the input buffers that it needs to. The generated image is collected in a single output buffer (since each pixel is only written once), and this is used to generate the output video stream. It is intended that the system would be implemented on FPGA using YHAC to generate the circuits from Ada source code, giving a truly parallel solution.

2.3 YHAC

The York Hardware Ada Compiler (YHAC) [4, 5, 6] allows Ada programs to be targeted directly to hardware. Using the SPARK subset and Ravenscar tasking profile gives a static language, which can be transformed to hardware. The compilation process uses template instantiation over the statements within the program. The templates build up to form a hardwired state-machine which controls the program flow Expressions are built up in a similar manner to produce expression trees. Complex expressions are split to allow multi-cycle evaluation. Concurrency is implemented using separate circuits, providing a truly parallel implementation. The only interference experienced by a task is over access to shared data.

In the domain of dependable and real-time systems, implementation via YHAC has several advantages:

- The produced circuit is traceable back to the source code.
- The program is implemented as a circuit, meaning there are no hardware bottlenecks, which need no architectural speed-up features.
- The final circuit can be easily analysed for resource usage. As the circuit is built up by template instantiation, analysis can be done from the source code. This covers both its space utilisation on the FPGA, and the timing of the program.
- Provides performance equivalent to a mid-range processor for single threaded applications.

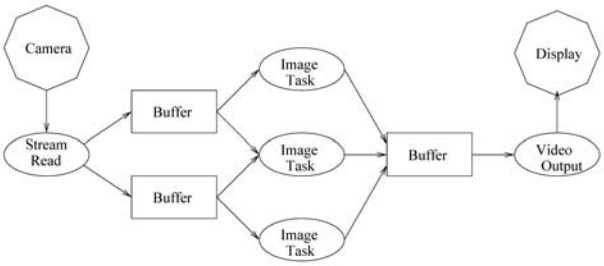

Figure 1 Diagram of the initial solution

- Concurrent applications get a significant performance boost due to parallelism. No longer sharing single processing resource reduces the level of inter-task interference.
- Designed to give the same semantics for all code in hardware as software. Ignores some implementation techniques (e.g. suspension objects) to maintain this consistency.

2.4 Solution

The solution presented above has several problems:

- There is no broadcast semantic in Ada. This makes filling multiple buffers difficult. Cannot broadcast the data to multiple POs, which needs more time per input pixel, but the timing of the video stream is fixed.
- There is no way to simultaneously release multiple tasks. Whilst entry queues allow multiple releases, these cannot happen simultaneously (each task has to enter the PO in turn). The Ravenscar profile exacerbates this problem as it outlaws entry queues.
- Working with video streams requires accurate timing in the circuit to ensure no pixels are lost. Whilst YHAC allows timing properties to be determined, it doesn't give definite control over the timing.

These problems require some changes to the original solution. By including dedicated hardware to interface with the video-streams, the lack of definite timing in YHAC is no longer an issue. This hardware can also handle some of the image pre-processing required, such as conversion of the data into RGB format, and clipping the input stream to the visible area. The buffers are also encapsulated within the hardware as this removes the need for a broadcast within Ada, and the structure of the compiler prevents the sharing of memory used in the buffers between dedicated hardware and Ada circuit. One advantage of encapsulating the buffers is that the accesses can be pipelined, improving their performance, allowing a smaller number of buffers to be used. The resulting change to the structure of the solution can be found in figure 2.

2.5 Implementation

The final system solution was implemented using a Celoxica RC203e development board. This board provides a Xilinx Virtex2 3000 FPGA as the logic resource, and a

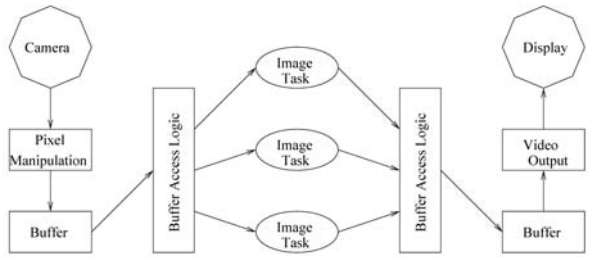

Figure 2 Diagram of the final solution

wide selection of interfaces, including video input and output. The board also provides 4MB of off chip memory. The drivers for all the interfaces are provided in Handel-C, Celoxica's C-based hardware language.

The implementation can be divided into two parts: the framework, which implements the video interfacing and buffering; and the application, which contains the transformation encoding. The implementation of these parts is described below:

- *Framework*
  As the device drivers are written in Handel-C, the dedicated hardware has been implemented in the same language. The these read in the video stream into multiple double buffers. Once a frame has been put into the buffers, the buffers are swapped, and a signal given to the application to start processing on the frame. Whilst this is happening, the next frame is being placed into the other set of buffers. On the output side, the transmission of a frame waits for the previous frame to finish, at which point the buffers are swapped and the new frame started. This double buffering introduces a delay of 1 frame plus the delay in the application. This cannot be reduced if full frame transformations are being dealt with. It can be seen that the maximum application delay is 1 frame - if it is slower, frames will only be part complete when transmission commences. There is no synchronisation between the input and output streams, so there is no additional cost over the delay of the application.
- *Application*
  The example implementation is an image warper. This takes a good image, and distorted it to give a fish-eyed image. Due to restrictions on the memory capacity on the board, the image is restricted to a resolution of 640 x 480. The image processing is implemented in 9 parallel tasks, each of which is responsible for part of the image. There are a number of 'helper' POs in the system, an interrupt handler PO for each task which detects the start processing signal from the framework and release the tasks, and a finish detector which provides the signal to the framework. Each task implements a simple transformation, which is pre-computed to save time in the processing. The transformation is done in under 1/2 frame, giving a delay of 1 1/2 frames overall.

The complete system took about a month to design and implement. The framework took most of this time, mostly in altering the provided sequential access buffers mostly in altering the provided sequential access buffers to allow the random access needed by the applications, and integrating the Handel-C and Ada circuits. The application took about 1 day to implement, half for application coding, the other half generating an acceptable transform. Due to the tool-chains needed in targeting hardware, the compile-test-correct cycle can take a while (a small value change still requires a complete re-compilation and synthesis), which extended the time needed to generate the transform. Alteration of the application is easy as the transform is coded algorithmically within the processing tasks.

The implemented system only uses 15% of the resources available on the FPGA, leaving plenty of scope for more complex transforms, or faster implementations. At present, most of the resource is taken by the framework, with the application itself using about 3%. By introducing more tasks, a faster implementation is possible, at the cost of higher resource usage. Alternatively extra resource can be used by making the transformation more complicated. If a lower resource usage is needed, the number of tasks can be reduced and the speed dropped to give a full frame's delay. The main limitation to the ultimate performance of the system, is the buffer throughput rate, but this can be increased by providing more buffers, at the cost of needing more memory buses on the device.

3 Language Suggestions

In designing the IMS, there appeared a need for two extra bits of functionality in the Ada language: a broadcast semantic, and a parallel release semantic. These are described below:

3.1 Broadcast

The broadcast semantic would allow a task to write data to a set of protected objects in a single call.

It is envisioned that the protected objects being targeted would be declared as an array of protected objects. This would allow existing array syntax to be used for the declaration of the POs, for selection of POs within the call, and permits a subset of the array to be selected. This gives the suggested syntax as shown in figure 3.

Two alternatives are presented for the broadcast to all elements of the array. The first uses the reserved word all to indicate that the entire array is being referenced, the second uses a slice that covers all elements of the array. A third, though discounted option would use the others keyword. These have their advantages and disadvantages:

All – The reserved word all in the name.all context is an explicit dereference of an access type. To use the same syntax here would overload it to be a reference to all elements within the array. There is also the problem of what happens when the array is accessed via an access type. However, the use of all does convey the meaning to the programmer that the entire array is being accessed.

```
1  protected type po_type is
2    procedure call(val : integer);
3  end p type;
4
5  po_array : array (1..10) of po_type;
6
7  po_array(7).call(37);       - - single instance call
8  po_array.all.call(25);      - - broadcast to all
                                  elements of po array
9  po_array(1..10).call(25); - - alternative broadcast
                                  to all elements of po array
10 po_array(2..5).call(13);   - - broadcast to restricted
       range of elements
```

Figure 3

Whole slice – Using an entire range slice maintains consistency in the selection of the parts of the array to use. There is no change in the structure of the selection, simplifying the compiler implementation. However, there is poor readability as it is not possible to tell that the entire array is being accessed. Similarly, if the array size is changed in a program, every whole slice will need to be changed, which does not aid program maintenance.

Others – The others keyword could also be used to indicate all elements in an array. This would follow from its use in aggregate expressions, but does not sit well out of the aggregate form. In addition, it does not have the same readability as the other forms.

On balance, the name.all form seems to offer the better balance, providing an obvious indication that the entire array is being accessed.

The implementation of the broadcast semantic can fit easily into both concurrent and parallel system implementations. Within a concurrent system, the calls to the protected procedures in the broadcast can be done by iterating through the array. In a parallel system, all the accesses can be initiated in parallel, provided sufficient processing elements are available. When there aren't enough processors there needs to be some iteration over the calls.

The envisioned implementation raises a couple of issues with semantics of the call. Since the calls may be done iteratively, to preserve the atomicity of the operation, there are two conditions that must hold. First, the call must be none-blocking. As the calls are being done iteratively, a block will delay the later calls. Secondly, there should be no pre-emption between calls, that is, the entire access should be considered a single protected action.

3.2 Guarded Protected Function

A guarded protected function semantic would allow multiple, read-only, accesses to wait on a guard value. When the guard becomes true, all accesses are allowed to enter immediately. On completion of these accesses the guard is automatically reset to false. These threads must be read only, as multiple threads would be active in the protected object. This preserves the access rules for protected actions and effectively gives a function based equivalent of an entry.

Since a guarded protected function is effectively a function based entry, a mix of the current function and entry call syntax would seem appropriate. A suggested syntax is shown in figure 4. The function specification follows that of a normal entry, with the addition of the return value type specification before the guard. There are two restrictions on the specification: the parameters to the entry can only be of in mode; and the guard expression must be a single boolean variable. The body of the entry will follow the rules of both functions and entries: no side-effects, that is, no change of PO state including the guard expression; no potentially blocking operations; and there must be a return statement in all paths through the body. The no side-effects rule prevents the guard from being reset inside the function, and hence the need for the automatic reset of the guard.

The calling of the guarded protected function remains the same as any other function call. There is an issue with this syntax in that forcing the entry call to be a function call may not reflect its use. In cases where it is used to allow multiple tasks to collect the same data on release, the use of the function call syntax is sensible. Where the only purpose is to effect a simultaneous release of multiple tasks (as in the example) the return value is not needed, but must still be used. Whilst this can be ignored by a compiler (a constant return value can be implemented as a local assignment after the call), it reduces the readability (and elegance) of the program code. However, making the call a procedure call would change the declaration syntax (and need a new reserved keyword to describe it) and require that out mode parameters be allowed. This is to allow data to be returned from the call, which also means that assignment to local parameters needs to be permitted, making the no side-effects rule harder to enforce.

An alternative syntax to that proposed in figure 4 would make no change to the syntax of the language. As mentioned above, the guarded function is a function-based equivalent of an entry. By changing the concept from a function call to a parallel entry call, it can be implemented without a syntax change. Restricting an entry to have only out mode parameters, a simple guard variable, and no side-effects would allow parallel access to it. The requirement for such an entry to be used as a parallel entry could be indicated using a pragma. This pragma would indicate to the compiler that this entry could be accessed in parallel, that the entry needed to be checked for conformity to the above requirements, and that the guard variable needed to be automatically reset. Whilst this does not require additional language syntax, it overloads the entry syntax with a different semantic, which could cause issues with readability and maintainability.

Again, this new semantic can be implemented in both parallel and concurrent systems. Within a parallel system, the readers are allowed access as soon as the protected procedure that set the guard to true completes. Since they are only reading the data within the PO, this can be done

```
1  protected type po_type is
2    function par_entry return integer when allow_entry;
3    procedure release;
4  private
5    allow_entry : boolean := false;
6  end po_type;
7
8  protected body po_type is
9    function par_entry return integer when allow_entry is
10   begin
11     return 0;
12   end par_entry;
13   procedure release is
14   begin
15     allow_entry := true;
16   end release;
17 end po_type;
18
19 task waiter is
20   null : integer
21 begin
22   while true
23     null := par_entry;
24     - - do something
25    end while;
26 end waiter;
```

Figure 4

without violating the protection rules. Once the final reader has left the PO, and the protected action completes, the guard value is reset to false (hence the requirement that it be a single variable). The concurrent implementation cannot have all the accesses happening at once, so they must be allowed to happen one after the other, all within the same protected action, and with the completion of the last access causing the guard variable to be reset to false. In this way it behaves in similarly to a ”last one out closes the door” implementation on an entry queue. In both these implementations, the resetting of the guard occurs as a result of the completion of the protected action that sets it. This allows for a simple definition of the semantics.

Both the suggestions for the guarded protected function semantic have used an automatic return of the guard to false after all the waiting calls have completed. This can be considered poor deign, as it hides some semantics of the call. An alternative solution would be to leave the resetting of the guard variable to the programmer. This can be easily accomplished using an entry guarded by the count attribute of the guarded function. Leaving this to the programmer will give a greater flexibility, at the cost of leaving open the possibility of bugs caused by programming errors.

3.3 Other Thoughts

Both of these suggestions are related to protected objects. In full Ada (as opposed to Ravenscar Ada), entries exist in both tasks and protected objects. This raises the question of whether the broadcast and parallel entry semantics be extended to tasks.

It would seem that a task broadcast would be a useful semantic to have. This would allow data to be broadcast directly to tasks, rather than forcing the use of POs between the tasks. However, the PO broadcast semantics use protected procedure calls, whereas the tasks only offer entries. Protected procedure calls, though subject to possible delay, are deemed to be non-blocking. Task entries are, however, deemed to be blocking and therefore provide a different semantic. As an entry call can only proceed when the task allows it, the broadcast can be held up by a single non-responsive task.

A task can only have a single thread of control. This makes a parallel entry impossible to implement as the calls would have to be handled serially as for a normal entry queue. In this case, accepting the first call would require that all were handled without interruption until the queue was empty. Of course, both of these would be outside the scope of the Ravenscar profile.

Finally, it should be noted that both the parallel entry and the broadcast can be emulated by the other, though with restrictions on the effectiveness of the emulation. A broadcast could be programmed through the parallel entry semantic, with the data to be broadcast being written to the PO and the waiting tasks allowed to read it. This provides a broadcast to the waiting task, any task that was wot waiting when the write happened would never be able to access that data, and would have to wait for the next broadcast. Similarly, the parallel entry can be emulated by broadcasting to multiple POs, each of which has a task waiting on an entry. This would allow each task to release once. However, the release time could not be guaranteed - in the parallel entry, only those tasks waiting get released; in the broadcast the task will release on the broadcast, or when it next tries to access, rather than being forced to wait for the next broadcast.

4 Conclusions

This paper has looked at the issues surrounding using Ada to implement a parallel stream based system. The problems were illustrated through the development of an Ada based image manipulation system. As a result of this, two suggestions for new language semantics as a result of problems encountered were presented.

The image manipulation system, developed to meet a set of industrial requirements, uses Ada to implement a stream based, parallel, image morpher. Whilst the implemented solution provides a framework for efficient image processing, it highlighted two shortcomings in the Ada language. There is no facility for easily splitting an input stream into several buffers, nor is there the ability to simultaneously release multiple tasks.

From the problems encountered in the implementation of the system, two new semantics have been proposed. The broadcast semantic will allow the same data to be written to multiple protected objects at the same time. The guarded protected function semantic provides a function based equivalent of an entry with the ability to release multiple

tasks simultaneously. Together, these would provide better support for parallel streaming applications.

References

[1] Ada 95 Reference Manual. Intermetrics, January 1995.

[2] J. Barnes. High Integrity Ada: The SPARK Approach. Addison-Wesley, 1997.

[3] A. Burns, B. Dobbing, and G. Romanski. The Ravenscar Tasking Profile for High Integrity Real-Time Programs. In Reliable Software Technologies, Proceedings of the Ada Europe Conference, Uppsala, volume 1411, pages 263–275. LNCS, Springer-Verlag, 1998.

[4] M. Ward and N. C. Audsley. Hardware Compilation of Sequential Ada. In Proceedings of CASES 2001, pages 99–107, 2001.

[5] M. Ward and N. C. Audsley. Hardware Implemanation of the Ravenscar Tasking Profile. In Proceedings of CASES 2002, pages 59–68, 2002.

[6] M. Ward and N. C. Audsley. Hardware Implementation of Programming Languages for Real-Time. In Proceedings of RTAS 2002, pages 276–285, 2002.

Ada Gems

The following contributions are taken from the AdaCore Gem of the Week series. The full collection of gems, discussion and related files, can be found at http://www.adacore.com/category/developers-center/gems/.

Ada Gem #19 — XML streaming of Ada objects

Pascal Obry, EDF R&D

Date: 26 November 2007

Let's get started…

Since Ada 95 it has been possible to stream any object. Using 'Input/'Output or 'Read/'Write attributes, any object (tagged or not) can be streamed using a binary representation. This means that objects can be written into a file or sent over a socket, for example.

Let's take a simple object hierarchy to illustrate this feature. We'll have a Point (x and y coordinate) and a Pixel (a Point with a color).

```
package Object is

   type Point is tagged record
      X, Y : Float;
   end record;

   type Color_Name is (Red, Green, Blue);

   type Pixel is new Point with record
      Color : Color_Name;
   end record;
end Object;
```

When writing a Point or a Pixel the first bytes in the stream are the tag external representation, and then the object's attribute values.

```
declare
   File : File_Type;
   P    : Point'Class := ...;
begin
   Create (File, Out_File, "streamed.data");
   Point'Class'Output (Text_Streams.Stream (File), P);
   Close (File);
end;
```

The stream will contain something like (where is the character hexadecimal code):

<01> <00> <00> <00> <0C> <00> <00> <00> O B J E C T . P I X E L <9A> <99> <99> <3f> <66> <66> <06> <41> <00>

The tag is an important part as it will be used to be able to create the proper object instance out of the stream.

```
P := constant Point'Class :=
     Point'Class'Input (Text_Streams.Stream (File));
```

All is well! No, there is a little missing feature. There is no way to control how the external tag is streamed. In fact, it is a string and the bounds (lower and upper) are first output into the stream. These bounds are plain numbers written in binary.

In the above example we have the four first bytes for lower bound (equal to 1) and the four following bytes for the upper bound (equal to 12) then the twelve bytes for the external tag full name OBJECT.PIXEL.

In Ada 95 there is no way to stream a textual representation of objects!

But the good news is… Ada 2005 can do this. Ada 2005 goes further by adding support to control finely the external representation of any objects. This means that it is now possible to create a textual representation of such an object using the 'Class'Input and 'Class'Output attributes.

Let's put in place the missing pieces.

First the 'Read and 'Write attributes to output or read the XML representation of a Point or a Pixel.

```
with Ada.Streams;

package Object is

   type Point is ...

   procedure Read (S : access Root_Stream_Type'Class;
                   O : out Point);
   for Point'Read use Read;

   procedure Write (S : access Root_Stream_Type'Class;
                    O : in Point);
   for Point'Write use Write;

   type Pixel is ...

   procedure Read (S : access Root_Stream_Type'Class;
                   O : out Pixel);
   for Pixel'Read use Read;

   procedure Write (S : access Root_Stream_Type'Class;
                    O : in Pixel);
   for Pixel'Write use Write;
```

The Read routines could be implemented using a full featured XML parser like XML/Ada. For conciseness, we will use two very simple XML oriented routines:

```
procedure Skip_Tag
   (S      : access Ada.Streams.Root_Stream_Type'Class;
   Ending : in     Character := '>');
- - Skip the next tag on stream S, returns
   when Ending is found
```

```
function Get_Value
  (S : access Ada.Streams.Root_Stream_Type'Class)
   return String;
- - Returns the current value read on stream S
```

Using those routines the 'Read and 'Write implementation are straightforward. Here is the implementation for a Point:

```
procedure Read (S : access Root_Stream_Type'Class;
                O : out Point) is
begin
   Skip_Tag (S); O.X := Float'Value (Get_Value (S));
   Skip_Tag (S, ASCII.LF);
   Skip_Tag (S); O.Y := Float'Value (Get_Value (S));
   Skip_Tag (S, ASCII.LF);
end Read;

procedure Write (S : access Root_Stream_Type'Class;
                 O : in Point) is
begin
   String'Write (S, "   <x>"  & Float'Image (O.X) &
                 "</x>" & ASCII.LF);
   String'Write (S, "   <y>"  & Float'Image (O.Y) &
                 "</y>" & ASCII.LF);
end Write;
```

The last missing piece is the handing of the tag. We want the tag to be simply: <point> and <pixel> (no bound and just the name of the object instead of the full name prefixed by the enclosing package name). To set the proper tag name we use the External_Tag attribute:

```
package Object is
   type Point is ...
   for Point'External_Tag use "point";

   type Pixel is ...
   for Pixel'External_Tag use "pixel";
```

Then we want to plug in our own XML oriented implementation of the 'Class'Input and 'Class'Output attributes. This is necessary only for the root type Point:

```
package Object is

   type Point is ...
   for Point'External_Tag use "point";

   procedure Class_Output
       (S : access Ada.Streams.Root_Stream_Type'Class;
        O : in Point'Class);
   for Point'Class'Output use Class_Output;

   function Class_Input
       (S : access Ada.Streams.Root_Stream_Type'Class)
     return Point'Class;
   for Point'Class'Input use Class_Input;
```

The Class_Output routine must output the opening XML tag, output the object itself and then the closing XML tag. Quite simple to do; the following is the commented code:

```
procedure Class_Output
     (S : access Ada.Streams.Root_Stream_Type'Class;
      O : in Point'Class) is
begin
   - - Write the opening tag <tag_name>
   Character'Write (S, '<');
   String'Write (S, Ada.Tags.External_Tag (O'Tag));
   String'Write (S, '>' & ASCII.LF);

   - - Write the object, dispatching call to Point/Pixel'Write
   Point'Output (S, O);

   - - Write the closing tag </tag_name>
   String'Write (S, "</");
   String'Write (S, Ada.Tags.External_Tag (O'Tag));
   String'Write (S, '>' & ASCII.LF);
end Class_Output;

function Class_Input
     (S : access Ada.Streams.Root_Stream_Type'Class)
   return Point'Class
is
   function Dispatching_Input is new
        Ada.Tags.Generic_Dispatching_Constructor
           (T => Point,
            Parameters  =>
            Ada.Streams.Root_Stream_Type'Class,
            Constructor => Point'Input);
     input     : String (1 .. 20);
     input_Len : Natural := 0;
begin
   - - On the stream we have <tag_name>,
   - - we want to get "tag_name"
   - - Read first character, must be '<'
   Character'Read (S, Input (1));
   if Input (1) /= '<' then
      raise Ada.Tags.Tag_Error with "Starting with " &
            Input (1);
   end if;

   - - Read tag
   Input_Len := 0;
   for I in Input'range loop
      Character'Read (S, Input (I));
      Input_Len := I;
      exit when Input (I) = '>';
   end loop;

   - - Check ending tag
   if Input (Input_Len) /= '>'
     or else Input_Len <= 1
   then - - Empty tag
      raise Ada.Tags.Tag_Error with "empty tag";
   else
      Input_Len := Input_Len - 1;
   end if;

   declare
      External_Tag : constant String :=
           Input (1 .. Input_Len);
      O : constant Point'Class :=
             Dispatching_Input (
             Ada.Tags.Internal_Tag (External_Tag), S);
```

```
      - - Dispatches to appropriate
      - - Point/Pixel'Input depending on
      - - the tag name.
    begin
      - - Skip closing object tag
      Skip_Tag (S, ASCII.LF);
      return O;
    end;
  end Class_Input;
```

At this point the code shown at the start will still work without modification. The fact that the object is streamed using an XML representation is transparent to the users of the Object package.

As a final note, for conciseness, the code as-is does not output conformant XML documents as there is no XML header and there are multiple root nodes. This is left as an exercise to the reader.

Ada Gem #21: How to parse an XML text

Emmanuel Briot, AdaCore

Date: 10 December 2007

Abstract: The World Wide Web Consortium (W3C) develops various specifications around the XML file format. In particular, it specifies various APIs to load, process and write an XML file. Although these APIs are not specified for Ada, XML/Ada tries to conform as closely as possible to them. This gem describes how to use XML/Ada to parse an XML file.

Let's get started…

There are two main APIs to parse an XML file. One (the Document Object Model, DOM) reads the file and generates a tree in memory representing the whole document. Typically, because of the amount of operations mandated by the specifications, this tree is several times larger than the document itself, and thus depending on the amount of memory on your machine, it might limit the size of documents your application can read. On the other hand, it provides a lot of flexibility in the handling of these trees.

The other method (SAX) is based on callbacks, which are called when various constructs are seen while reading the XML file. This requires almost no memory, but makes the processing of the XML file additional work for your application. It is however very well suited when you want to store the XML data in an application-specific data structure. In fact, XML/Ada itself uses SAX to build the DOM tree.

In both cases, XML/Ada needs an object (an "input_source") to read the actual XML data. This data can be found either on the disk, in memory, read from a socket, or any other possible source you can imagine. XML/Ada is carefully constructed so that it doesn't require the whole document in memory, and can just read one character at a time, which makes it adaptable to any possible input. This gem does not cover how to write your own input streams. This is in general quite easy, the only difficulty is to properly convert the bytes you are reading to unicode characters.

Here is a small example on using the DOM API to create a tree in memory. In this example, we are assuming the most frequent case of an XML file on the disk, and therefore we are using a File_Input as the input. The second object we need is the XML parser itself. When we want to create a DOM tree, we need to use a Tree_Reader, or a type derived from it. As we will see later, this is in fact a SAX parser (that is an event-based XML parser) whose callbacks are implemented to create the DOM tree. You can of course override its primitive operations if you want to do additional things (like verbose output, redirect error messages, pre-processing of the XML nodes,…).

```
with Input_Sources.File;  use Input_Sources.File;
with DOM.Readers;         use DOM.Readers;
with DOM.Core;            use DOM.Core;

procedure Read_XML_File (Filename : String) is
  Input  : File_Input;
  Reader : Tree_Reader;
  Doc    : Document;
begin
  Open (Filename, Input);
  Parse (Reader, Input);
  Close (Input);

  Doc := Get_Tree (Reader);
  ...
  Free (Reader);
end Read_XML_File;
```

The first three lines read the file into memory. The fourth line gets a handle on the tree itself, which you can then manipulate with the various subprograms found in the DOM.Core.* packages (and that are mandated by the W3C specifications). When we are done, we simply free the memory.

There are various settings that can be set on the reader before we actually parse the XML stream, for instance whether it should support XML namespaces, whether we want to validate the input, and so on.

As we mentioned before, there exists a second, lower-level API called SAX which is event-based. It defines one tagged type, a Reader, which has several primitive operations that act as callbacks. You can override the ones you want. In general, the result of calling them is to create an in-memory representation of the XML input (which is what the DOM interface does, really).

The following short example only detects the start of elements in the XML file, and prints their name on standard output. It has little interest in real applications, but is a good framework on which to base your own SAX parsers.

```
with Sax.Attributes;
with Sax.Readers;     use Sax.Readers;
with Unicode.CES;     use Unicode.CES;

package Debug_Parsers is
   type Debug_Reader is new Reader with null record;
   overriding procedure Start_Element
     (Handler       : in out Debug_Reader;
      Namespace_URI : Unicode.CES.Byte_Sequence := "";
      Local_Name    : Unicode.CES.Byte_Sequence := "";
      Qname         : Unicode.CES.Byte_Sequence := "";
      Atts          : Sax.Attributes.Attributes'Class);
end Debug_Parsers;
```

Here is the implementation of the Start_Element callback. We are assuming, in this simple example, that the console on which we are printing the output can accept unicode characters (in fact, all Put_Line does is to print a series of bytes, which are interpreted by the console to do the proper rendering of unicode glyphs).

```
with Ada.Text_IO;   use Ada.Text_IO;

package body Debug_Parsers is
   procedure Start_Element
     (Handler       : in out Debug_Reader;
      Namespace_URI : Unicode.CES.Byte_Sequence := "";
      Local_Name    : Unicode.CES.Byte_Sequence := "";
      Qname         : Unicode.CES.Byte_Sequence := "";
      Atts          : Sax.Attributes.Attributes'Class)
   is
   begin
      Put_Line ("Found start of " & Qname);
   end Start_Element;
end Debug_Parsers;
```

And finally here is a short example of a program using that parser. Notice how it closely mimics what we did for DOM (which is not so surprising, since, once again, the DOM parser itself is really a special implementation of a SAX parser).

```
with Input_Sources.File;  use Input_Sources.File;
with Debug_Parsers;       use Debug_Parsers;

procedure Test_Sax is
  Input  : File_Input;
  Reader : Debug_Reader;
begin
  Open (Filename, Input);
  Parse (Reader, Input);
  Close (Input);
end Test_Sax;
```

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o K.U. Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Peter Dencker
Steinäckerstr. 25
D-76275 Ettlingen-Spessartt
Germany
Email: dencker@web.de
URL: ada-deutschland.de

Ada-France

Association Ada-France
c/o Jérôme Hugues
Département Informatique et Réseau
École Nationale Supérieure des Télécomunications
46, rue Barrault
75634 Paris Cedex 135
France
Email: bureau@ada-france.org
URL: www.ada-france.org

Ada-Spain

attn. José Javier Gutiérrez
Ada-Spain
P.O.Box 50.403
28080-Madrid
Spain
Phone: +34-942-201-394
Fax: +34-942-201-402
Email: gutierjj@unican.es
URL: www.adaspain.org

Ada in Sweden

attn. Rei Stråhle
Saab Systems
S:t Olofsgatan 9A
SE-753 21 Uppsala
Sweden
Phone: +46 73 437 7124
Fax: +46 85 808 7260
Email: Rei.Strahle@saabgroup.com
URL: www.ada-i-sverige.se

Ada in Switzerland

attn. Ahlan Marriott
White Elephant GmbH
Postfach 327
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: ada@white-elephant.ch
URL: www.ada-switzerland.ch