ADA USER JOURNAL

Volume 30
Number 2
June 2009

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Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.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

I am writing this Editorial in the aftermath of the Ada-Europe 2009 conference in Brest, France As usual, the conference attracted a large number of Ada practioners, for a full week of workshops, tutorials, paper presentations, and, very important, fruitful informal interactions. This success of the conference is very valuable for the Ada community, and I am sure that it will be repeated, or even improved, next year in Valencia, Spain.

From the very rich program of the week, and apologizing in advance to the readers, I would like to point out the interesting poster exhibition provided during the conference, with the topic "Thirty Years of the Ada User Journal" (based on the article with the same name published in the March issue of the Journal). And I am happy to announce that the posters will also be exhibited at the SIGAda conference, next fall.

As for this second issue of volume thirty, it starts with the information of the News and Calendar sections, by Marco Panunzio and Dirk Craeynest, their respective editors. The forthcoming events section provides details on the 2009 SIGAda conference, which will take place next November in the Tampa Bay area, Florida, USA, and on the 15th International Conference on Reliable Software Technologies – Ada-Europe 2010 that will take place June 2010 in Valencia, Spain. The issue also provides a brief report on the Ada UK Conference 2009, which took place last March, in London, UK, by Kathy Fairlamb of AdaCore, France.

The technical part of the issue starts with a contribution from the Industrial Track of the Ada-Europe 2009 conference. John Harbaugh, of the Boeing Company, USA, presents details of an effort to improve part of a large Ada 95 Command, Control and Communications system, using design patterns and refactoring.

Afterwards, the reader will find summaries of three of the tutorials that were provided at the Ada-Europe 2009 conference. The first, by Quentin Ochem, of AdaCore, France, addressed multi language programming, and how to interface Ada with native languages, such as C or C++, and languages running in virtual machines, such as Java or Python. The second tutorial, provided by Robert Davis and Ian Broster, of Rapita Systems, UK, addressed the always difficult issue of execution time measurements and how to obtain accurate worst-case execution time estimates. In the third tutorial, Pat Rogers, of AdaCore, USA, detailed how to specify software fault tolerance mechanisms in Ada. Finally, the issue ends with two interesting Ada Gems, by Pat Rogers and Bob Duff, related to Stream I/O and Overload Resolution.

Luís Miguel Pinho
Porto
June 2009
Email: lmp@isep.ipp.pt

FOR IMMEDIATE RELEASE

Thirty years of the Ada User Journal

BREST, France (June 11, 2009) – On the occasion of Ada-Europe 2009, the 14th annual Conference on Reliable Software Technologies, Ada-Europe, the international organization that promotes the knowledge and use of Ada in European academia, research and industry, inaugurated the celebrations of the 30th Anniversary of the Ada User Journal.

Ada User Journal, the quarterly publication of Ada-Europe, keeps its readership abreast of developments in the standardization, use and promotion of the Ada programming language and technology, as well as issues related with reliable software technologies and engineering in Europe and the rest of the world. The Journal also currently maintains an on-line accessible archive of past editions since early 2002.

The origins of the Ada User Journal date back to the birth of *Ada UK News*, which started publication in March 1980. As the first Editor, Prof. Ian Pyle, at the time Chair of Computer Science at the University of York, UK, put it in his inaugural Editorial, "*Ada has elicited interest in both the industrial and academic worlds. Perhaps for the first time we can find ourselves pulling in the same direction to the benefit of us all. This is an opportunity for bridging the industrial/academic gulf which we must not lose*". And in fact this vision is still one of the cornerstones of the Journal Editorial policy.

The current name of the Journal first appeared in Volume 15 in the year 1994, when it was still published by Ada UK. Ada-Europe published the *Ada-Europe News* since June 1989, until it was merged with the Ada User Journal in March 1998. From that time onward, Ada-Europe and Ada UK jointly published the Journal until Ada-Europe took over as the sole publisher from Volume 23 in 2002.

To mark this anniversary celebration, the March 2009 issue of the Ada User Journal features a special article entitled "Thirty years of the Ada User Journal", which recalls its three decades of history. Celebratory posters were also exhibited at the Ada-Europe 2009 conference. A special issue of the Journal, to be released in March 2010, will reprint a selection of the best articles published in the Journal over the past 30 years.

About Ada-Europe
Ada-Europe is the international non-profit organization that promotes the knowledge and use of Ada into academia, research and industry in Europe. Current member organizations of Ada-Europe are: Ada-Belgium, Ada in Denmark, Ada-Deutschland, Ada-France, Ada-Spain, Ada in Sweden and Ada-Switzerland. Ada-Europe also includes and welcomes individual members from other European countries with no national organization, and has a total membership in the region of 300.

A PDF version of this press release is available at www.ada-europe.org.

Press contact
Dirk Craeynest, Ada-Europe Vice-President, Dirk.Craeynest@cs.kuleuven.be

News

Marco Panunzio

University of Padua. Email: panunzio@math.unipd.it

Contents

Ada-related Organizations

ARA — ACATS 3.0J and 3.0K

From: Ada Information Clearinghouse
Date: Mon, 30 Mar 2009
Subject: Ada Conformity Assessment Test Suite
URL: http://www.adaic.com/whatsnew.html

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

From: Ada Information Clearinghouse
Date: Thu, 9 Apr 2009
Subject: Ada Conformity Assessment Test Suite
URL: http://www.adaic.com/whatsnew.html

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

[see also "ARA — ACATS 3.0E" in AUJ 29-3 (Sep 2008), p.149 —mp]

Ada 2005 R2 Language Reference Manual

From: Ada Information Clearinghouse
Date: Fri, 20 Mar 2009
Subject: Ada 2005 R2 Language Reference Manual
URL: http://www.adaic.com/standards/ada1z.html

The documents on this page consolidate a possible second amendment to Ada 95 with the previously standardized Amendment 1, Technical Corrigendum 1, and the Ada Standard (International Standard ISO/IEC 8652:1995).

The Amendment (Amendment 2) will be produced by the ISO/IEC JTC 1/SC 22/WG 9 Ada Rapporteur Group (ARG). The final form of Amendment 2, or whether its standardization will succeed, are not known at this time. Thus, any proposed feature may be substantially changed or withdrawn before the Amendment begins standardization.

These draft documents are not an official publication or work product of the ARG, but rather are provided by the ARA as a service to the Ada community. The intent is that this will be a modest update to the Ada language, and thus the language will continue to be known as Ada 2005 after Amendment 2 is completed (rather than Ada 2012 or some similar name). To differentiate it from the existing language when that is necessary for comprehensibility, we are calling it Ada 2005 R2 (Ada 2005 Release 2).

For more on the possible amendment, see the ARG working site.

[http://www.ada-auth.org/amendment2.html —mp]

The current (Ada 2005) consolidated standard is available here.

[http://www.adaic.com/standards/ada05.html —mp]

This is draft 7. This version contains AIs that were ARG-approved through the February 2009 ARG meeting, along with some presentation issues (AI05-0092-1).

Send editorial comments on the documents to agent@ada-auth.org. Editorial comments are those that do not change the meaning of the text, such a spelling errors, doubled words, etc. Substantive comments should be submitted to the Ada-Comment mailing list (ada-comment@ada-auth.org) as outlined in Introduction of the draft standard.

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. —mp]

Ada-Belgium 2009

From: Dirk Craeynest <dirk@aqua.cs.kuleuven.be>
Date: Sat, 2 May 2009 14:57:37 +0200 CEST
Subject: Ada-Belgium Spring 2009 Event, incl. Debian packaging workshop
Newsgroups: comp.lang.ada, fr.comp.lang.ada

Ada-Belgium Spring 2009 Event

Sunday, May 17, 2009, 12:00-19:00

Leuven, Belgium

including at 14:00

2009 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 17, 2009 in Leuven.

Last year, 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. As the new format was appreciated by all those present, this year's event once more starts at noon, runs until 7pm, and includes a barbecue, a key signing party, the 16th 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 12, to Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be> with the subject "Ada-Belgium Spring 2009 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 2009 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 2009 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-form09.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 sum of 10 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/09/090517-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 CDROM distribution.

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

[8] http://people.debian.org/~lbrenta/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!

Dirk Craeynest

President Ada-Belgium

Dirk.Craeynest@cs.kuleuven.be

Acknowledgments

We would like to thank our sponsors for their continued support of our activities: AdaCore, Katholieke Universiteit Leuven (K.U.Leuven), Offis nv/sa - Aubay Group, and Université Libre de Bruxelles (U.L.B.).

Ada-Europe 2009 Call for Paricipation

From: Dirk Craeynest <dirk@aqua.cs.kuleuven.be>
Date: Sun, 19 Apr 2009 23:16:19 +0200 CEST
Subject: 14th Int.Conf.on Reliable Software Technologies, Ada-Europe 2009
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

Call for Participation

*** PROGRAM SUMMARY ***

14th International Conference on

Reliable Software Technologies – Ada-Europe 2009

8 - 12 June 2009, Brest, France

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

Organized by Ada-Europe,

in cooperation with ACM SIGAda

Early registration discount until May 15

Ada-Europe organizes annual international conferences since the early 80's. This is the 14th event in the Reliable Software Technologies series, previous ones being held at Montreux, Switzerland ('96), London, UK ('97), Uppsala, Sweden ('98), Santander, Spain ('99), Potsdam, Germany ('00), Leuven, Belgium ('01), Vienna, Austria ('02), Toulouse, France ('03), Palma de Mallorca, Spain ('04), York, UK ('05), Porto, Portugal ('06), Geneva, Switzerland ('07), Venice, Italy ('08).

The Advance Program brochure with full information will shortly be available on the conference web site. It will contain the list of accepted papers, industrial and educational presentations, as well as detailed descriptions of tutorials and keynote presentations.

Also check the conference web site for registration, accommodation and travel information.

Quick overview

- Mon 8 & Fri 12: tutorials, workshops
- Tue 9 - Thu 11: paper, industrial & vendor presentations, exhibition

Proceedings

- published by Springer-Verlag
- volume 5570 in Lecture Notes in Computer Science series (LNCS)
- will be available at conference

Invited speakers

- John Benito, Blue Pilot Consulting, USA, "ISO JTC1/SC22/WG23 Work on Programming Language Vulnerabilities"
- Pierre Sens, LIP6, Université Pierre et Marie Curie, Paris, France, "Fault Tolerance in Large Scale Distributed Systems"
- Peter H. Feiler, Software Engineering Institute, Carnegie Mellon University, USA, "Validation of Safety-Critical Systems with AADL"

Tutorials (full day)

- "Building Cross Language Applications Using Ada", Quentin Ochem, AdaCore, France
- "SPARK - the Libre Language and Toolset for High-Assurance Software", Roderick Chapman, Praxis High Integrity Systems, UK

Tutorials (half day)

- "An Introduction to Parallel and Real-Time Programming with Ada", John McCormick, University of Northern Iowa, USA
- "Software Fault Tolerance", Pat Rogers, AdaCore, USA
- "Software Measures for Building Dependable Software Systems", William Bail, MITRE, USA
- "Modeling for Schedulability Analysis with the UML Profile for MARTE", Julio Medina, Universidad de Cantabria, Spain, and Huascar Espinoza, CEA-List, France
- "Hard Real-Time and Embedded Systems Programming", Pat Rogers, AdaCore, USA
- "Designing Real-Time, Concurrent, and Embedded Software Systems using UML and Ada", Rob Pettit, The Aerospace Corporation, USA
- "Object-Oriented Programming in Ada 2005", Matthew Heaney, On2 Technologies, USA
- "Execution Time: Analysis, Verification, and Optimization in Reliable Systems", Ian Broster, Rapita Systems, UK

Workshops (full day)

- "Software Vulnerabilities"
- "AADL"

Papers and Presentations

- 19 refereed technical papers in sessions on High-Integrity, Testing, Education, Real-Time, Model-Driven Engineering, MDE and AADL, Ensuring Software Integrity
- 6 industrial presentations on current practice and challenges
- submissions by authors from 19 countries, and accepted contributions from Argentina, Australia, China, France, Italy, Spain, Switzerland, UK, and USA

Exhibition

- 5 exhibitors already committed: AdaCore, Aonix, Ellidiss Software, IBM, and Rapita Systems; others expressed interest
- vendor presentation track for exhibitors

Social evening events

- Tuesday: welcome reception at Oceanopolis, Brittany's sea park by the Marina in Brest, including a guided tour, concert and buffet
- Wednesday: conference banquet by the sea side in the charming village of Porspoder, located 25 km northwest of Brest

Registration

- early registration discount up to Fri May 15, 2009
- additional discount for academia, Ada-Europe and ACM SIGAda members
- registration includes copy of printed proceedings at event
- includes coffee breaks and lunches
- three day conference registration includes social events
- payment possible by bank transfer or credit card

We recommend all participants to book hotel accommodation as soon as possible, as numerous events are organized in June in Brest.

For more info and latest updates see the conference web site at http://www.ada-europe.org/conference2009.html or contact the local chair at ae2009-reg@mlistes.telecom-bretagne.eu.

[…]

IRTAW-14 Call for Papers

From: Dirk Craeynest <dirk@cs.kuleuven.ac.be>
Date: Sat, 25 Apr 2009 05:47:25 +0000 UTC
Subject: IRTAW-14 Call for Papers, International Real-Time Ada Workshop
Newsgroups: comp.lang.ada

Posted per request of the organizers.

Please note the deadline is about 2 weeks from now: Friday 8 May.

[…]

Workshop announcement - Call for Papers

IRTAW-14

14th International Real-Time Ada Workshop, 2009

Portovenere, Italy

7-9 October 2009

Organized in cooperation with Ada-Europe

Deadline for Papers: 8th May 2009.

See workshop web site for details of the Call.

For over 20 years the series of International Real-Time Ada Workshop meetings has provided a forum for identifying issues with real-time system support in Ada and for exploring possible approaches and solutions, and has attracted participation from key members of the research, user, and implementer communities worldwide. Recent IRTAW meetings have significantly contributed to the Ada 2005 standard, especially with respect to the tasking features, the real-time and high-integrity systems annexes, and the standardization of the Ravenscar profile.

Workshop Web site:
http://events.math.unipd.it/irtaw14/

Hotel Location :
http://www.royalsporting.com/index.php?&id_lang=1

Neil Audsley PC Chair

neil@cs.york.ac.uk

Safety-Critical Embedded Software Testing Symposium

From: Vector Software Press Center
Date: Thu, 26 Mar 2009
Subject: Vector Software to host a Safety Critical Embedded Software Testing Symposium
URL: http://www.vectorcast.com/pdf/press-release-for-symposium.pdf

East Greenwich, RI – March 26, 2009.

Vector Software, the leading provider of software test tools for embedded systems, today announced that it will host the Safety Critical Embedded Software Testing Symposium on Thursday, May 14, 2009 from 8:30 a.m. to 12:00 p.m. EST.

The Symposium will focus on best practices to reduce time, effort and cost in validating your embedded software.

The event will take place at the Forefront Center in Waltham, MA. Further information can be found on the Vector Software web site at

http://www.vectorcast.com/news/newsletters/symposium.html

The Safety Critical Embedded Software Testing Symposium is a must attend event for those looking for answers to their software testing challenges.

Topics to be discussed include:

- Best practices to meet FAA, FDA and other regulatory agency requirements
- How to optimize your code to ensure testing success
- How some of our customers accomplish their testing objectives
- Via a comprehensive live demonstration on an embedded target, how VectorCAST, the industry-leading embedded software testing tool, addresses unit testing, integration testing, code coverage analysis, system testing, and regression testing

About Vector Software

Vector Software, Inc. is the leading independent provider of automated software testing tools for developers of safety-critical embedded applications. Vector Software's VectorCAST line of products, automate and manage the complex tasks associated with unit, integration, and system level testing.

The VectorCAST tools support the C, C++, and Ada programming languages.

[…]

Review of Ada Issues

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Date: Mon, 13 Apr 2009 14:42 +0200
Subject: Review of Ada Issues for June 2009 SC22/WG9 meeting (fwd)
Mailing list: ada-belgium-info@cs.kuleuven.ac.be

Dear Ada-Belgium friend,

The following message was just posted to the Ada-Belgium members' mailing list and is reposted here for your information.

[…]

Dear Ada-Belgium member,

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 2009 conference next June in Brest, France.

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

AI05-0003-1/03 2008-11-14 - Qualified expressions and "names"

AI05-0009-1/09 2009-03-10 - Confirming rep. clauses and independence

AI05-0050-1/06 2008-11-18 - Return permissions are not enough for build-in-place

AI05-0054-2/05 2008-11-26 - Variable views of constant objects

AI05-0067-1/08 2008-11-18 - Objects that are built in place

AI05-0069-1/03 2007-11-26 - Holder container

AI05-0095-1/03 2008-11-19 - Address of intrinsic subprograms

AI05-0099-1/03 2008-11-19 - The tag, not the type, of an object determines if it is controlled

AI05-0100-1/03 2008-11-19 - Placement of pragmas

AI05-0101-1/04 2008-11-13 - Remote functions must support external streaming

AI05-0103-1/04 2009-03-09 - Return statements should require at least static compatibility

AI05-0106-1/03 2008-11-20 - Representation items are not allowed on generic formal parameters

AI05-0108-1/02 2008-11-18 - The incomplete view from a limited view does not have a discriminant part

AI05-0109-1/02 2008-11-18 - Impossible check in S'Class'Input

AI05-0112-1/02 2009-03-09 - Names for anonymous aspects of representation

AI05-0116-1/02 2008-11-20 - The value of Alignment for a class-wide object

AI05-0118-1/02 2008-11-17 - How do parameter associations associate?

AI05-0120-1/02 2008-11-18 - The current instance of a protected object is a constant view

AI05-0126-1/02 2008-11-14 - Dispatching when there is no declared operation

AI05-0128-1/02 2009-03-09 - "/=" is a primitive operation

AI05-0129-1/03 2009-03-09 - A limited view does not contain views of incomplete types

AI05-0132-1/02 2009-03-09 - A library unit pragma must apply to a library unit

AI05-0133-1/02 2009-03-09 - Extending a type with a self-referencing discriminant constraint on a component

AI05-0134-1/02 2009-03-09 - Full conformance should include the profiles of anonymous access-to-subprogram

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, Wednesday, May 13th, 2009.

Early comments are encouraged.

Dirk Craeynest

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

Dirk.Craeynest@cs.kuleuven.be

[…]

Ada and Education

"Ada for Software Engineers" textbook

From: Ada Information Clearinghouse
Date: Wed, 27 May 2009
Subject: Ada 2005 version of Ben-Ari's Ada textbook
URL: http://www.adaic.com/whatsnew.html

An Ada 2005 version of Ben-Ari's popular Ada textbook has been published.

Ada for Software Engineers (Second Edition with Ada 2005)

Prof. Mordechai (Moti) Ben-Ari, Department of Science Teaching, Weizmann

Institute of Science

http://stwww.weizmann.ac.il/g-cs/benari

ISBN-13: 978-1-84882-313-6

"Ada for Software Engineers" teaches the language as it is used in practice through relatively large case-studies such as a discrete event simulation.

The presentation emphasizes the features for object-oriented and systems programming that were introduced in Ada 95, as well as the new features in Ada 2005. A graduated introduction to the terminology and style of the language reference manual makes this an ideal textbook for practicing software engineers.

AdaCore — Introducing SPARK Pro Webinar

From: AdaCore InSight Webinar Series
Date: Tuesday, 21 Apr 2009
Subject: Introducing SPARK Pro
URL: http://www.adacore.com/home/products/gnatpro/webinars/

The InSight webinar series continued with a presentation by Rod Chapman on the AdaCore/Praxis new joint offering - SPARK Pro. SPARK Pro combines the proven SPARK Ada language and supporting toolset with AdaCore's GNAT Programming Studio (GPS) integrated development environment, backed by unrivaled support systems. SPARK is a language specifically designed to support the development of software used in applications where correct operation is vital either for reasons of safety or security. The SPARK Toolset offers static verification that is unrivalled in terms of its soundness, low false-alarm rate, depth and efficiency. The toolset also generates evidence for correctness that can be used to build a constructive assurance case in line with the requirements of industry regulators and certification schemes.

This webinar presents the concepts behind the Correctness-by-Construction methodology and includes a demo of the SPARK Pro toolset.

Ada-related Resources

SPARK Proof

From: JP Thornley <jpt@diphi.demon.co.uk>
Date: Thu, 19 Mar 2009 12:13:57 +0000
Subject: ANN: SPARK Proof - Tutorials and Tools
Newsgroups: comp.lang.ada

I have put a couple of tutorials on SPARK proof onto www.sparksure.com; one for proof of absence of run-time error (i.e. using the optional proof annotations) and one for using the Proof Checker. Both tutorials contain several sections, with worked examples and exercises.

I have also updated a couple of tools that help with SPARK proof:

1) VC_View makes it easier to read and interpret SPARK verification conditions.

2) PCHIF is an interface to the Proof Checker that makes it easier to recall and edit previously entered commands and to control the commands that are saved. (Previous versions of PCHIF were very unstable, but this is now sorted, thanks to Dmitry Kazakov and Maxim Reznik for their Gtk Router.)

The tutorials and the tools (Windows executables only at present) can be downloaded from www.sparksure.com.

Tokeneer — Update of proofs

From: Phil Thornley <phil.jpthornley@googlemail.com>
Date: Mon, 27 Apr 2009 02:59:57 -0700 PDT
Subject: ANN: Tokeneer - Proofs updated to use User Rules
Newsgroups: comp.lang.ada

The Tokeneer code is an excellent example of SPARK, but the work was completed several years ago and prior to major improvements being made to the proof capabilities of the SPARK Toolset. Consequently the published example does not fully demonstrate those capabilities nor does it provide examples of how to use them.

I have now revised many of the proofs to use User Rules, with a reduction in unsimplified VCs from 110 to 24. The Proof Checker is not now required to complete any of the proofs (although it is still used to prove VCs that justify two of the rules).

The files needed to update the published version of Tokeneer are available from www.sparksure.com. There is a note included with the files describing the changes made and the approach used.

From: Roderick Chapman <roderick.chapman@googlemail.com>
Date: Wed, 29 Apr 2009 00:20:39 -0700 PDT
Subject: Re: ANN: Tokeneer - Proofs updated to use User Rules
Newsgroups: comp.lang.ada

Nice work Phil. We're planning to release an updated Tokeneer package later this year following the GPL release of the SPARK Toolset, so we'll try to include these if that's OK. What licence are your new rules under?

From: Phil Thornley <phil.jpthornley@googlemail.com>
Date: Thu, 30 Apr 2009 04:10:43 -0700 PDT
Subject: Re: ANN: Tokeneer - Proofs updated to use User Rules
Newsgroups: comp.lang.ada

I'll be more than happy for you to include them in an updated Tokeneer release.

> What licence are your new rules under?

The files are supplied for anyone to use as they want, with no restricitions on how they can be used. I'll sort out actual terms for your copies by email.

Book on Ada 95 by Jean-Pierre Rosen

From: Pascal Pignard <sur.pignard@wanadoo.fr>
Date: Sun, 10 May 2009 18:46:00 CEST
Subject: Méthodes de génie logiciel avec Ada
Mailing list: ada-france.ada-france.org

The book "Méthodes de génie logiciel avec Ada 95" by Jean-Pierre Rosen, printed in June 1995 in the wake of the Ada 95 standard and now out of printing, has found a new life on Wikibooks.

The idea is to place it in the Internet in collaborative mode in order to extend and enrich it. I naturally thought to Wikipedia and Wikibooks altough it is my first attempt to use them as a contributor.

I hope many of you will appreciate the text by Jean-Pierre Rosen that I tried to reproduce as faithfully as possible, and will contribute to its enrichment.

Foreword: A quick overview of the Ada language

http://fr.wikibooks.org/wiki/Méthodes_de_génie_logiciel_avec_Ada

[translated from French —mp]

Ada-related Tools

New homepage for RAPID

From: Oliver Kellogg <okellogg@freenet.de>
Date: Sat, 21 Mar 2009 14:57:26 -0700 PDT
Subject: ANN: RAPID has a new home
Newsgroups: comp.lang.ada

The Rapid Ada Portable Interface Designer project is being revived and has its new home at:

http://savannah.nongnu.org/projects/rapid/

The Tcl/Tk and Gtk backends have been updated to newer toolkit versions. The other backends (JGNAT, .NET, GWindows) are not currently maintained. Contributors are very welcome.

The Subversion repository is at

http://svn.savannah.nongnu.org/svn/rapid/trunk/

Thanks go to Martin Carlisle, the original RAPID author, for his cooperation.

From: Jerry Bauck <lanceboyle@qwest.net>
Date: Sun, 22 Mar 2009 14:28:58 -0700 PDT
Subject: Re: ANN: RAPID has a new home
Newsgroups: comp.lang.ada

Glad to see this. How about a couple of screen shots (current link on project web site is old and dead) and a simple code example or two?

From: Oliver Kellogg <okellogg@freenet.de>
Date: Mon, 23 Mar 2009 00:13:18 -0700 PDT
Subject: Re: ANN: RAPID has a new home
Newsgroups: comp.lang.ada

It's in the making. Please check back in a week or so.

(Right now I'm concentrating on the core RAPID implementation, see svn trunk.)

RAPID 3.2

From: Oliver Kellogg <okellogg@users.sourceforge.net>
Date: Mon, 30 Mar 2009 07:53:08 +0200
Subject: RAPID 3.2 is released
Newsgroups: comp.lang.ada

After a lengthy pause since the last version, an update to the Rapid Ada Portable Interface Designer has been released and is available at

http://savannah.nongnu.org/files/?group=rapid

Here are the news for RAPID 3.2:

- Tested with Tcl/Tk version 8.{4,5,6a} and TASH version 8.6-0
- Tested with gtk+-2.{10,12,14} and GtkAda from svn trunk of Jan 2009
- GUI file format maintains backward compatibility with RAPID 3.01
- New command line switch -ni supports non-interactive Ada code generation for a given .gui file without launching the RAPID GUI
- New widget: FRAME, for creating an empty space with a border
- New widget: TEXTBOX, similar to TEXTENTRY but for read-only text, supports optional user variable of type Ada.Strings.Unbounded.Unbounded_String
- TEXTENTRY now also supports unsigned user types
- CHECKBUTTON optional user variable can now be any two-valued enum type in addition to Boolean
- The procedures Fill_Window, Generate_and_Fill_Window, Read_Window are not created if they are null operations. In other words, they are only created if there are user variables to read or write.
- The JGNAT, .NET, and GWindows peers are not included as they are not currently maintained.
- This release is only tested on Linux and it does not contain a Windows exe.

For more info, see

http://savannah.nongnu.org/projects/rapid/

Milter API

From: Björn Persson <bjorn@xn--rombobjrn-67a.se>
Date: Sat, 18 Apr 2009 18:35:49 +0200
Subject: Ann: Milter API
Newsgroups: comp.lang.ada

I needed to write a milter to expel the spammers from my mailbox, and naturally I wanted to write it in Ada. I couldn't find a binding to Libmilter so I made one. I present the first public version of the Ada Milter API:

http://www.rombobjörn.se/Milter_API

The binding is in a "works for me" state. It's written to match the version of Libmilter that comes with Sendmail 8.13.8, because that's what I currently have on my server. It provides almost all of the features in that version of Libmilter, but a few pieces are missing because I don't use them and they weren't trivial to implement. Later versions of Libmilter also have new features that the binding doesn't support yet.

I use Milter API with Postfix 2.3.8 and it's working great, but functions that I don't use are pretty much untested. I haven't even tried it with Sendmail, because I use Postfix.

There is also a binding to the syslog functions, which Milter API uses:

http://www.rombobjörn.se/System_Log

Thanks in part to Milter API, I no longer need to obfuscate my email address.

From: Albrecht Käfer <albrecht_kaefer@yahoo.de>
Date: Fri, 24 Apr 2009 17:40:32 +0200
Subject: Re: Ann: Milter API
Newsgroups: comp.lang.ada

> […] Those links appear to be broken -- they're not working for me, anyway.

Then your browser doesn't support international URLs. Use these:

http://www.xn--rombobjrn-67a.se/Milter_API

http://www.xn--rombobjrn-67a.se/System_Log

Ada binding to GMP and MPFR

From: Vincent Diemunsch <vincent.diemunsch@gmail.com>
Date: Wed, 20 May 2009 07:19:09 -0700 PDT
Subject: Ada binding to GMP and MPFR
Newsgroups: comp.lang.ada

[…]

As the libraries GMP (http://gmplib.org/) and MPFR (http://www.mpfr.org/) becomes part of the GNAT free compiler (since they are part of the new GCC) and since these libraries have excellent performances, I thought it could be interesting to create an Ada binding for them.

I searched on Internet and found two old binding but non convinced me and they seemed to have been dropped out. Therefore, I have undertaken to write my own binding for GMP and MPFR in Ada 2005, and I have just managed to successfully test basic operations for them such, as adding or multiplying MPFR_Floats of different precision…

Now I wonder if someone could be interested in hosting the sources on an Internet page, so that anybody could download them and test them and eventually make improvements…

The Binding is as follows :

- A THIN BINDING composed of two files :
  - o gmp.ads
  - o mpfr.ads

It basically translates in Ada most functions of gmp.h and mpfr.h but it's not exhaustive.

There are also some specific files for target dependent types:

- o mp_x86_32bits.ads

o mp_x86_64bits.ads...

- A THICK BINDING with the following specification files:
  - o gmp.Integers.ads;
  - o gmp.Rationals.ads;
  - o mpfr.Floats.ads;

These files declare the following types:

Unbounded_Integer, Unbounded_Fraction, MPFR_Float

that can be seen as extensions of the typical Ada Integer and Float, with operator overloading "+", "*", ... and elementary functions.

I know that some work needs to be done to:

- complete the gmp.ads and mpfr.ads files
- fully test the binding
- have a good integration in the Ada "spirit".

Therefore, I require your help:

- Where may i found a place to put the files (or a SVN repository ;-)
- Would you be interested in giving comments on implementation especially for MPFR_Types ?

[…]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 20 May 2009 07:30:24 -0700 PDT
Subject: Re: Ada binding to GMP and MPFR
Newsgroups: comp.lang.ada

[…]

I can offer hosting on the Ada-France monotone server[1,2] but I think it would be more productive if you would assign your copyright to the FSF[3] and submit your binding for inclusion in GCC. That way, your bindings would become part of GCC along with the libraries themselves and would be packaged along with GCC into most distributions. This requires that you use the same license, i.e. GPL version 3 or later with an additional clause permitting use in proprietary programs.

[1] http://www.ada-france.org/article130.html (French version)

[2] http://www.ada-france.org/article131.html (English version)

[3] http://gcc.gnu.org/contribute.html

From: qunying <zhu.qunying@gmail.com>
Date: Wed, 20 May 2009 08:24:34 -0700 PDT
Subject: Re: Ada binding to GMP and MPFR
Newsgroups: comp.lang.ada

[…]

What Ludovic suggested is the best way. In case you don't want that route, as you are using a gmail account, I think the most easy route for you is to host it in

http://code.google.com/projecthosting/

as long as it uses the selected few open sources licenses, no pre-approval is required. The drawback is that it only support svn so far, and the Mercurial support is still not open to all.

Others required approval from the hosting site:

http://sourceforge.net/

http://savannah.gnu.org/

http://github.com/

[…]

From: Vincent Diemunsch <vincent.diemunsch@gmail.com>
Date: Wed, 27 May 2009 08:35:44 -0700 PDT
Subject: Re: Ada binding to GMP and MPFR
Newsgroups: comp.lang.ada

Thanks for your response. I have tried to host in GOOGLE Source:

http://code.google.com/p/adabindinggmpmpfr/

I will now continue to test it. I would be very interested to recieve comments on it:

- on the specification, if it is easy to use and clear,
- on the implementation,

[…]

From: John B. Matthews <nospam@nospam.invalid>
Date: Wed, 27 May 2009 13:47:43 -0400
Subject: Re: Ada binding to GMP and MPFR
Newsgroups: comp.lang.ada

[…]

Somewhat tangentially, on Mac OS X, GNAT from MacAda looks for MPFR and GMP in /usr/local/lib. Users may wish to know that the libraries may be conveniently built from source using MacPorts:

http://www.macports.org/

http://trac.macports.or/browser/trunk/dports/devel/mpfr/Portfile

http://trac.macports.org/browser/trunk/dports/devel/gmp/Portfile

http://www.macada.org/

See also:

http://trac.macports.org/browser/trunk/dports/lang/gnat-gcc/Portfile

From: Vincent Diemunsch <vincent.diemunsch@gmail.com>
Date: Fri, 29 May 2009 05:01:52 -0700 PDT
Subject: Re: Ada binding to GMP and MPFR
Newsgroups: comp.lang.ada

[…]

I have put the license FSF version 3 on all the files and created a project in:

http://code.google.com/p/adabindinggmpmpfr/

Now I would like to test it a bit more before submitting it to GCC.

[…]

APQ 3.0 Beta1

From: Marcelo C. de Freitas <marcelo.batera@gmail.com>
Date: Wed, 11 Mar 2009 18:37:02 -0700 PDT
Subject: APQ 3.0-b1 released
Newsgroups: comp.lang.ada

[…]

Just to announce the 3.0-b1 release for APQ. It can be downloaded from:

http://adaworks.net/releases/apq-3.0-b1/

This is the first beta release since we took over APQ maintenance and it features:

- build system fully based on gpr files
- each database driver has it's own folder
- bug fixes
- Under a new license :: MGPL (the same license for both closed and open sourced projects)

Big thanks to:

Adèle Helena Ribeiro

Adrian-Ken Rueegsegger

Alex Abate Biral

Jesse Lang

Peter C. Chapin

who have contributed to this release. And thanks also Warren W. Gay who have made APQ a reality in the first place. We are just trying to give our best to make this an even better database binding for Ada.

Go there, download, test and submit bug reports please!

For the curious ones, the plans for the next release includes:

- ODBC support (it's already implemented but build system is broken and doesn't support unixODBC yet)
- handling data types by their primitives instead of string type

From: Marcelo C. de Freitas <marcelo.batera@gmail.com>
Date: Wed, 11 Mar 2009 18:59:46 -0700 PDT
Subject: Re: APQ 3.0-b1 released
Newsgroups: comp.lang.ada

I forgot to mention:

- Microsoft SQL Support in the new apq-ct_lib module
- both apq-ct_lib and apq-sybase can be compiled against FreeTDS as an alternative to Sybase's library.

Various Ada scripts

From: Oliver Kellogg <okellogg@users.sourceforge.net>
Date: Thu, 02 Apr 2009 04:08:07 +0200
Subject: ANN: Updates to Ada related scripts
Newsgroups: comp.lang.ada

[…]

Updates are available for ada2idl.pl and adareps2c.pl at

http://freenet-homepage.de/okellogg/x.html

- ada2idl.pl version 0.4 fixes conversion of record components
- adareps2c.pl version 0.3 fixes placement of __attribute__((packed))

Valgrind 3.4.1

From: Valgrind Webpage
Date: Sat, 28 Feb 2009
Subject: Release of Valgrind 3.4.1
URL: http://valgrind.org/docs/manual/dist.news.html

Release 3.4.1 (28 February 2009)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

3.4.1 is a bug-fix release that fixes some regressions and assertion failures in debug info reading in 3.4.0, most notably incorrect stack traces on amd64-linux on older (glibc-2.3 based) systems. Various other debug info problems are also fixed. A number of bugs in the exp-ptrcheck tool introduced in 3.4.0 have been fixed.

In view of the fact that 3.4.0 contains user-visible regressions relative to 3.3.x, upgrading to 3.4.1 is recommended.

Packagers are encouraged to ship 3.4.1 in preference to 3.4.0.

The fixed bugs are as follows. Note that "n-i-bz" stands for "not in bugzilla" -- that is, a bug that was reported to us but never got a bugzilla entry. We encourage you to file bugs in bugzilla (http://bugs.kde.org/enter_valgrind_bug.cgi) rather than mailing the developers (or mailing lists) directly -- bugs that are not entered into bugzilla tend to get forgotten about or ignored.

n-i-bz Fix various bugs reading icc-11 generated debug info

n-i-bz Fix various bugs reading gcc-4.4 generated debug info

n-i-bz Preliminary support for glibc-2.10 / Fedora 11

n-i-bz Cachegrind and Callgrind: handle non-power-of-two cache sizes, so as to support (eg) 24k Atom D1 and Core2 with 3/6/12MB L2.

179618 exp-ptrcheck crashed / exit prematurely

179624 helgrind: false positive races with pthread_create and recv/open/close/read

134207 pkg-config output contains @VG_PLATFORM@

176926 floating point exception at valgrind startup with PPC 440EPX

181594 Bogus warning for empty text segment

173751 amd64->IR: 0x48 0xF 0x6F 0x45 (even more redundant rex prefixes)

181707 Dwarf3 doesn't require enumerations to have name

185038 exp-ptrcheck: "unhandled syscall: 285" (fallocate) on x86_64

185050 exp-ptrcheck: sg_main.c:727 (add_block_to_GlobalTree): Assertion '!already_present' failed.

185359 exp-ptrcheck unhandled syscall getresuid()

(3.4.1.RC1: 24 Feb 2008, vex r1884, valgrind r9253).

(3.4.1: 28 Feb 2008, vex r1884, valgrind r9293).

Ada-related Products

AdaCore / Praxis — SPARK Pro

From: AdaCore Press Center
Date: Tuesday February 17, 2009
Subject: Praxis and AdaCore Announce SPARK Pro
URL: http://www.adacore.com/2009/03/24/spark-pro/

NEW YORK, PARIS and LONDON, March 24, 2009 - Ada UK Conference - Developers creating safety critical and high assurance systems will benefit from today's launch of SPARK Pro. The new open source development environment has been created by Praxis, international specialist in critical systems engineering, and AdaCore, the leading provider of commercial software solutions for the Ada language.

SPARK Pro combines the proven SPARK language and supporting toolset with AdaCore's easy-to-use GNAT Programming Studio (GPS) Integrated Development Environment, backed by unrivalled support services. This provides a powerful method for developing critical systems.

Developed by Praxis, SPARK is a language specifically designed to support the development of software used in applications where correct operation is vital either for reasons of safety or security. The SPARK Toolset offers static verification that is unrivalled in terms of its soundness, low false-alarm rate, depth and efficiency. The toolset also generates evidence for correctness that can be used to build a constructive assurance case in line with the requirements of industry regulators and certification schemes. There are versions of SPARK based on Ada 83, Ada 95, and Ada 2005, so all leading Ada compilers and tools work out-of-the-box with SPARK.

The new development environment will be globally available from AdaCore, with support delivered by both companies using AdaCore's web based GNAT Tracker support system. Existing SPARK users have the option to transition to the new environment, which is also available as a standalone product. The launch of SPARK Pro is the first available product following the technical and marketing partnership announced in Q4 2008 between Praxis and AdaCore.

"SPARK continues to be used for the development of new, advanced, critical systems as well as support of existing operational systems, so it is vital that it continues to develop – SPARK Pro is a major step forward in usability and support through our partnership with AdaCore," said Keith Williams, Praxis Managing Director. "The launch of SPARK Pro integrates the strengths of the best development tools in the market. Our partnership combines our proven expertise with AdaCore's user interface, global reach and support systems to deliver the perfect solution for safety and security developers. This is great news for the SPARK language and our existing and future customers."

"AdaCore and Praxis share a commitment to enabling high-quality, high-integrity development," said Robert Dewar, AdaCore President and CEO. "I am excited by the possibilities of this partnership. The SPARK technology is an outstanding demonstration of the fact that formal methods technologies are practical today, and hold enormous promise for the future. The launch of SPARK Pro, which integrates SPARK Pro and GNAT Pro into a coherent FLOSS tool set and environment, extends the benefits of SPARK to existing and new GNAT Pro users.

I expect this combination to play a major role in high-integrity development, especially for the increasingly important security-critical area".

"Praxis has worked with AdaCore for many years and the development of common technology was the next logical step," said Sylvain Haman, Praxis Director.

"SPARK Pro will enable our clients world-wide to ensure correctness of their software while enjoying the convenience of an enhanced development environment."

Praxis and AdaCore have a long history of working together on high profile systems. These include the Tokeneer project developed by Praxis for the US National Security Agency (NSA) using the SPARK Ada language and toolset and AdaCore GNAT Pro.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Webinar

A joint webinar between AdaCore and Praxis to discuss the benefits and provide a demonstration of SPARK Pro will be held on April 21, 2009. It will begin at 5pm European Daylight Time/4pm GMT Daylight Time/11am Eastern Daylight Time/8am Pacific Daylight Time. To register please visit

https://adacore.webex.com/adacore/onstage/g.php?t=a&d=718810241.

About Praxis

Praxis is a systems engineering company specializing in safety and mission critical applications. Praxis leads the world in specific areas of advanced systems engineering, such as ultra low defect software engineering, safety engineering for complex or novel systems, and tools/methods for systems engineering. Praxis offers clients a range of services including turn-key systems development, consultancy, training and R&D. Key market sectors are Aerospace, Defence, Air Traffic Management, Railways, Nuclear and Automotive.

The company operates internationally with active projects in North America, Asia and Europe. The headquarters of Praxis are in Bath (UK) with further offices in London, Loughborough, and Paris. It is wholly owned by Altran Technologies, which is a global leader in innovation engineering and has 18,500 staff across the world.

www.praxis-his.com

About AdaCore

Founded in 1994, AdaCore is the leading provider of commercial software solutions for Ada, the state-of-the-art 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 world-wide customer base; see

www.adacore.com/home/company/customers/

for further information.

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

AdaCore has North American headquarters in New York and European headquarters in Paris.

www.adacore.com

Press Contacts

press@adacore.com

Leena Chauhan

Praxis

leena.chauhan@praxis-his.com

AdaCore — Traceability Analysis Package for DO-178B

From: AdaCore Press Center
Date: Wednesday March 11, 2009
Subject: AdaCore Launches Traceability Analysis Package for DO-178B
URL: http://www.adacore.com/2009/03/11/adacore-launches-traceability-analysis-package-for-do-178b/

PARIS, NEW YORK and AMSTERDAM, March 11, 2009 – Avionics 2009 - AdaCore, leading provider of Ada tools and support, today announced the availability of the GNAT Pro Traceability Analysis Package. This product and services solution comprises an Ada language feature analysis, including test cases and GNAT Pro switch recommendations, that can help developers demonstrate compliance of safety-critical software with the DO-178B avionics standard.

Depending on the application's criticality level, DO-178B demands varying depths of analysis for showing coverage of the requirements by the software.

In general it is sufficient to demonstrate coverage based on the source code. However, at the highest level (DO-178B, Level A), if the compiler generates object code not directly traceable to source code, then the developer needs to perform additional verification on the object code to establish the correctness of such generated code. The GNAT Pro Traceability Analysis Package provides a product/services solution that can reduce this effort.

"For a system at DO-178B Level A, a failure could be catastrophic and cause the loss of human life," said Cyrille Comar, Managing Director, AdaCore Europe. "The release of the Traceability Analysis Package is a natural extension of our tool support for developers working in this critical area. It uses our intimate knowledge of both the Ada programming language and the GNAT Pro technology to provide accurate source-to-object code traceability analysis."

"The GNAT Pro Traceability Analysis Package allows developers to use richer subsets of Ada while reducing certification costs," said Robert Dewar, AdaCore President/CEO. "Although the code compiled for a more sophisticated feature might not be directly traceable to the source program construct, the analyses supplied in the Package provide the additional verification needed for compliance with DO-178B, Level A. The result is an overall reduction of effort and better usage of Ada language features."

The GNAT Pro Traceability Analysis Package includes the following items, which AdaCore prepares based on a safety-oriented coding standard supplied by the customer:

- Consistency analysis of the coding standard
- Recommendations for those compilation switches and language restrictions offering the best tradeoff between performance of generated code and ease of showing traceability between source and object code
- A test suite representative of the subset of the Ada language allowed by the customer's coding standard
- The analysis, for each test, of the traceability of the generated object code
- Additional verification to establish the correctness of generated code that is not directly traceable to source code.

The Traceability Analysis Package complements AdaCore's existing GNAT Pro High-Integrity Edition for DO-178B. This environment includes GNATcheck, a coding standard verification tool. With GNATcheck developers can enforce the specified coding standard or language subset covered by the Traceability Analysis Package.

Along with its partners, AdaCore is also heavily involved in "Project Coverage", the first Open Source code coverage project for DO-178B and safety-critical systems. "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. "Project Coverage" participants are AdaCore, Open Wide, ENST and LIP6 with financial support from French public funds.

About GNAT Pro

The GNAT Pro development environment, available on more platforms than any other Ada toolset, combines industry-leading technology with an expert support infrastructure and provides a natural solution for organizations that need to create reliable, efficient, and maintainable code. GNAT Pro is the first-to-market implementation of the Ada 2005 standard, allowing users to take advantage of the many enhancements in areas such as object-oriented programming, real-time support, and predefined libraries.

At the heart of GNAT Pro is a full-featured, multi-language development environment complete with libraries, bindings and a range of supplementary tools. All GNAT Pro technology is

distributed with complete source code. GNAT Pro is based on the widely used GCC technology, is subjected to a rigorous quality assurance process, and is backed by rapid and expert support service.

[…]

AdaCore — GNAT Pro for AVR

From: AdaCore Press Center
Date: Tuesday March 31, 2009
Subject: GNAT Pro available for 8-bit AVR Microcontroller
URL: http://www.adacore.com/2009/03/31/8-bit-avr-microcontroller/

SAN JOSE, Calif. and PARIS, March 31, 2009 - Embedded Systems Conference - AdaCore, provider of the highest quality Ada tools and support, today announced availability of its GNAT Pro Ada development environment for the Atmel® AVR® 8-bit microcontroller. This brings the high-level language benefits of Ada to the embedded systems community who have requirements for low-power and small-memory devices, easing the job of developing safety-critical and high-security deeply embedded applications.

The first user of GNAT Pro for AVR will be the UK's Atomic Weapons Establishment (AWE). AWE is a long-standing Ada user, and GNAT Pro's support for the AVR processor will enable AWE to extend its Ada usage to new projects.

GNAT Pro for AVR benefits from the features and functionalities of GNAT Pro's existing High Integrity Edition. It comes with a Zero Footprint run-time library that is particularly well adapted to meet the small memory constraints of the AVR microcontroller. GNAT Pro for AVR demonstrates the suitability of the Ada programming language for safety-critical development for architectures ranging from 8-bit to 32-bit and 64-bit processors.

"The industry has long been waiting for an Ada solution on 8-bit microcontrollers. GNAT Pro for AVR demonstrates AdaCore's capability to answer the needs of customers who work in highly constrained environments," said Michaël Friess, Technical Sales Manager at AdaCore. "Organizations such as AWE that are using the AVR microcontroller will find a product that takes into account the increasing interest in formal methods, providing a natural fit with SPARK and Praxis High-Integrity Systems' correctness-by-construction approach."

Atmel's low-power, high-performance AVR microcontroller handles demanding 8-bit applications. With a single cycle instruction RISC CPU, innovative picoPowerTM technology, and a rich feature set, the AVR architecture ensures fast code execution combined with the lowest possible power consumption.

"Following an extensive evaluation, we decided to standardize on Ada as our development language of choice due to its early error detection and its support for our safety-critical, high-reliability requirements," said a spokesperson for the UK Atomic Weapons Establishment. "Since many of our projects include deeply embedded systems, GNAT Pro for AVR is an ideal solution."

[…]

AdaCore — GNAT GPL 2009

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Date: Thu, 28 May 2009 22:18:00 CEST
Subject: GNAT GPL 2009 available
Mailing list: ada-belgium-info @cs.kuleuven.be

Dear Ada-Belgium friend,

We hereby forward you an announcement from our long time corporate member and sponsor AdaCore about the new GNAT GPL 2009 release.

[…]

----- Forwarded message -----

Dear GNAT GPL user,

We are pleased to announce the release of GNAT GPL 2009, the Ada Toolset for Academic users and FLOSS developers. It introduces many new features including:

- Ability to generate byte code for the JVM
- Improved support for the .NET Framework
- Addition of the Ada-Java Interfacing Suite (AJIS) that enables native Ada code to be called from Java.
- Availability on the Mac OS X (64 bit) platform
- Automatic C/C++ binding generators
- Addition of the GNAT Component Collection (GNATcoll) providing new APIs that can be extended by the user community.

GNAT GPL 2009 comes with version 4.3.1 of the GNAT Programming Studio IDE and GNATbench 2.3, the GNAT plug-in for Eclipse.

It is available for the GNU Linux, Mac OS X (64 bit), .NET, JVM and Windows platforms.

GNAT GPL 2009 can be downloaded from the "Download" section on the new Libre website:

https://libre.adacore.com

[For AJIS, see http://www.adacore.com/2008/06/17/ada-java_interfacing_suite For GNATcoll, see http://www.adacore.com/2008/06/17/gnat_component_collection —mp]

AdaLog — AdaControl 1.11r3

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 13 Mar 2009 13:03:42 +0100
Subject: AdaControl 1.11r3 released
Newsgroups: comp.lang.ada

Adalog is pleased to announce the release of a new version of AdaControl.

This release includes small bug fixes, various improvements, and of course new rules, reaching 375 possible checks.

Small but useful improvement: if pfni (the utility that prints the full name of an entity) is called on a constant or a variable which is initialized to a static value, it prints that value after full static evaluation. This means that from GPS, you can right-click on an identifier, select "print full name", and get the fully evaluated initial value.

As usual, source and executable (GPL2008) versions are available for download from http://www.adalog.fr/adacontrol2.htm

[see also "AdaLog — AdaControl" in AUJ 29.4 (Dec 2008), p.237 —mp]

Lattix — Lattix 5.0

From: Lattix Press Center
Date: Thu, 28 May 2009
Subject: Lattix Releases Lattix 5.0
URL: http://www.lattix.com/news/articles/Lattix50.php

Award-winning software architecture management solution has powerful new capabilities for analyzing and re-architecting complex systems

Boston, MA–June 1, 2009– Lattix Inc., a leading provider of innovative software architecture management solutions, today announced the release of its newest solution, Lattix 5.0. This solution includes powerful new functionality to enable architects, developers and managers to identify issues, restructure and measure their system architecture.

In addition to performance improvements and feature enhancements, Lattix 5.0 contains very innovative new algorithms and architecture metrics. Also new in Lattix 5.0 are integrations to IBM Rational's Rhapsody for UML/SysML models and to Klocwork for C/C++ codebases.

"Our new algorithms and metrics provide not only the means to improve the modularity of your system and reduce defects, but also to measure and track improvements in the architecture" explains Neeraj Sangal, president and founder of Lattix. "Now it will be

possible for our customers to explore solutions to issues in their systems and assess the impact of proposed changes on the system quality."

"Our team at Conway had more than 230 projects in the Eclipse workspace which were required by every developer due to circular dependencies," explains Van Smity, technical lead at Conway Transportation. "Using Lattix's DSM partitioning algorithms, we optimized the projecct build order and our project sprawl is now free of circular dependencies. The new partitioning and reporting options available in Lattix 5.0 gives us greater visibility of system degredation and metrics to track the progress and effectiveness of future architectural decisions."

New capabilities in this major release include:

- Architecture Metrics: Lattix offers the most comprehensive set of "state of the art" architectural metrics. These metrics actually enable architects to pin-point complexity and identify architectural problems. Among the new metrics in Lattix 5.0 are ones which have a proven correlation to the defect rate of projects. Lattix also provides metric values for some of the leading industry projects allowing architects to compare how their own projects stack up to them.
- Partitioning Algorithms: Lattix now provides ten different partitioning algorithms to cover a broad range of use cases for the analysis and re-architecting of system elements. New sequencing and clustering algorithms facilitate grouping of elements even though they are cyclically coupled.
- System Reorganization Algorithms: These new algorithms can be used to restructure systems which lack sufficient structure or which can be improved by replacing the existing structure with a more modular one. Elements of selected subsystems can be redistributed automatically or through the use of labels applied by the user with the Lattix tagging capability.
- Compound Projects: It is now possible to create a compound project that consists of multiple individual projects. This enables the aggregation of projects while allowing each project to be maintained individually, which typically occurs in large projects involving numerous teams.

Also new to Lattix 5.0 is a direct integration with IBM Rational's Rhapsody. The Lattix for Rhapsody module, which is now part of the Lattix UML/SysML solution, provides a powerful systematic approach to review, refactor, and maintain architecture in large scale Rhapsody models. With Lattix for Rhapsody, a designer can readily identify undesirable interdependencies in the model which prevent modularity and increase complexity. As the model changes, Lattix for Rhapsody can be used to enforce the architecture and expose key design decisions for the entire team. For more information, please visit

http://www.lattix.com/products/LDMforRhapsody.php.

"Lattix integrates with Rhapsody to help teams improve the quality of their models and achieve enhanced productivity," said Michael Loria, vice president, Business Development, IBM Rational Software. "By using Lattix and Rhapsody, users are provided with an automated approach to managing software architecture that maximizes the value of their investment in IBM Rational software".

About Lattix 5.0

Lattix 5.0 provides the most comprehensive solution for systems that include UML/SysML models, codebases, databases, and frameworks. Lattix 5.0 supports XMI and IBM Rational Rhapsody models; Ada, C/C++, Java, .NET, and Pascal languages; Oracle, SQL Server, and Sybase databases; and Spring and Hibernate frameworks. Lattix 5.0 also provides support for full web-based reporting of architectural metrics, violations, and incremental changes. To learn more about Lattix 5.0 and explore the different solutions that are available, please visit

http://www.lattix.com/products/products.php.

Lattix 5.0 enables companies to improve and maintain quality, lower defect rates, enhance testability, lower costs through more effective development, and manage risks by better understanding of the impact of proposed changes.

Availability

Lattix 5.0 is available immediately from Lattix in the US or from our partners throughout Europe, the Middle East, and Asia Pacific. A variety of license options are available, from individual user to enterprise floating licenses. A free evaluation license is also available for download from

http://www.lattix.com/dl/gettingstarted.php.

About Lattix

Lattix is a leader of software architecture management solutions that deliver higher software quality and lower risk throughout the application lifecycle.

Lattix provides a powerful new approach of utilizing dependency models for automated analysis and enforcement of architectures.

Lattix is located in Andover, MA. More information about Lattix can be found at

www.lattix.com.

Midoan — Mika 1.1

From: info@midoan.com
Date: Mon, 25 May 2009 11:42:51 -0700 PDT
Subject: ANNOUNCE Mika by Midoan New Release : Automated Test Data Generation for Ada
Newsgroups: comp.lang.ada

Midoan Software Engineering Solutions Ltd. (http://www.midoan.com/) announces a new release of Mika, the first commercial testing tool for Ada that automatically generates test inputs from your source code.

Mika is an entirely automatic tool that analyses your Ada code and generates, carefully constructed, tests that will exercise all the branches or decision within your code at a level suitable for integration testing. With Mika, manual test data generation is no longer necessary.

Version 1.1 can be downloaded at

http://www.midoan.com/download.html

Of note in this new release is the possibility of using any GNAT compiler (including GNAT Pro).

Version 1.1 (25 May 2009)

- Improvements include:
  - o Mika is tool chain independent: any GNAT compiler can be used;
  - o Tests are generated even in the absence of test points;
  - o Error messages are more informative;
  - o GNAT's specific attributes are now compiler dependent and generated on the fly;
  - o Tests generation for exponent expressions with integer argument is more powerful;
  - o A direct uninstall is available;
- Subset enlargements include:
  - o Mika is now based on Ada 2005;
  - o Additional GNAT's specific attributes are appropriately handled (including storage_unit and word_size);
  - o Additional constructs are ignored rather than causing a failure (including Exception handlers);
  - o The Address predefined attribute returns a dummy value rather than fail;
  - o Qualified array expressions are now appropriately handled rather than causing a failure (array aggregates were already handled);
- Bug fixes include:
  - o Use of derived array and derived record types objects was sometimes causing errors;
  - o Zero exponent expression with integer argument did not always return 1;

o Qualified expressions were sometimes causing errors for integer, floats and enumeration types;

See http://www.midoan.com/ for further information including examples and documentation.

Ada and GNU/Linux

On big files in 32-bit and 64-bit systems

From: Olivier Scalbert <olivier.scalbert@algosyn.com>
Date: Mon, 25 May 2009 21:13:30 +0200
Subject: Large files on 32 and 64 bits ystem
Newsgroups: comp.lang.ada

[…]

I need to create a file that has 2540160000 bytes, that is a little more than 2**31 bytes.

On a 64 bits Linux box, it is ok, but on a 32 bits Linux box, I have:

raised Ada.IO_EXCEPTIONS.DEVICE_ERROR : s-fileio.adb:1135

The file length is: 2147483647, which is 2**31 - 1

Is it possible to write more than 2**31 bytes with an Ada program on a 32 bits Linux?

From: Dennis Lee Bieber <wlfraed@ix.netcom.com>
Date: Mon, 25 May 2009 15:19:04 -0700
Subject: Re: Large files on 32 and 64 bits ystem
Newsgroups: comp.lang.ada

Not an actual answer, but it may depend upon how the filesystem addressing was built - the above indicates 32-bit SIGNED offsets.

32-bit unsigned would permit 4GB files rather than 2GB.

Not sure what WinXP NTFS uses -- but since DV-AVI is 13GB/hour, and I've had many hours transferred, it must be using a 64-bit addressing even on a 32-bit machine.

The FAT filesystem used on memory cards tends to be limited to either 2GB or 4GB per file, even though the card may be much longer (a limitation encountered on those video cameras using memory card storage, and also some audio recorders when running long sessions and/or high resolution [96kHz, 24-bit] - which breaks the data into files at the 2GB point, often causing a loss as it takes a few seconds to close one file and start a new one)

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 25 May 2009 16:23:07 -0700 PDT
Subject: Re: Large files on 32 and 64 bits ystem
Newsgroups: comp.lang.ada

> […] raised Ada.IO_EXCEPTIONS.DEVICE_ERROR : s-fileio.adb:1135

That line indicates that the GNAT run-time library delegates the write to fwrite(3), so your question really boils down to whether the C run-time library has support for large files or not. What filesystem type do you use on your machines? I use XFS which supports files up to 8 exabytes :-)

> The file length is: 2147483647, which is 2**31 - 1

This limitation exists on FAT16 ("msdos" in Linux parlance). Newer filesystems have higher limits.

> Is it possible to write more than 2**31 bytes with an Ada program on a 32 bits linux ?

32-bit Linux has "large file support" using either a dedicated 64-bit API or the O_LARGEFILE flag supported in open(2) since glibc 2.2.

It might be a good idea to check how GNAT's run-time library deals with this API […].

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 26 May 2009 03:22:42 -0700 PDT
Subject: Re: Large files on 32 and 64 bits ystem
Newsgroups: comp.lang.ada

>[…] I use ext3.

In this filesystem, files can grow to 16 GB to 2 TB depending on block size.

[1] explains how large file support works in GNU/Linux. Basically, a C program supports LFS if it is compiled with the -D_FILE_OFFSET_BITS=3D64 preprocessor option on the command line. This changes the definition of fopen(3), fwrite(3) et al to use 64-bit file offsets instead of the default 32-bit file offsets.

Unfortunately, the GNAT run-time library directly imports these functions from glibc without any preprocessor in between, so is restricted to 32-bit file offsets, and so does not support large files.

It would be an interesting project for a beginning GCC hacker to implement LFS in libgnat. This would involve:

- wrapper functions in adaint.c that call fopen(3), fwrite(3), etc.
- compiling adaint.c with -D_FILE_OFFSET_BITS=3D64
- calling the wrappers instead of the glibc functions in System.File_IO et al. (it may be a little bit tricky to import them properly on all platforms, with and without LFS).

[1] http://www.suse.de/~aj/linux_lfs.html

From: Robert A Duff <bobduff@shell01.TheWorld.com>
Date: Tue, 26 May 2009 09:26:41 -0400
Subject: Re: Large files on 32 and 64 bits ystem
Newsgroups: comp.lang.ada

> Unfortunately, the GNAT run-time library directly imports these functions from glibc without any preprocessor in between, so is restricted to 32-bit file offsets, and so does not support large files.

Recent versions of GNAT have better support for large files.

I don't know if this has made it into any public versions yet, probably not.

From: Anonymous
Date: Tue, 26 May 2009 00:49:19 GMT
Subject: Re: Large files on 32 and 64 bits ystem
Newsgroups: comp.lang.ada

With GNAT:

Ada.Direct_IO uses a file indexes that has the type of Count or Positive_Count.

And these types are based on the positive range which is define as the size of a long_Integer which is normally in a 32-bit machine set to positive value of (2**63 - 1).

So it is possible from Ada, but the interface links between Ada and the OS may limit the size. Or in some cases it could be the OS or device-drivers that is limiting the file size.

But in looking at the routine System.FileIO.Write_Buf where the exception occurred and the Interfaces.C.Streams they both limit the file size to Standard'Address_Size or in the case of GNAT 32-bit version, to a positive range of 32-bit word aka (2GB -1).

So, the answer is:

On a 64-bit machine is limited to 2**64-1 file size and on a 32-bit machine is limited to 2**32-1 file size.

Ada Inside

Use of Ada in the Sentinel-1 satellite

From: AdaCore Press Center
Date: Tue, 26 May 2009
Subject: Astrium in the UK Selects GNAT Pro for Environmental Satellite System
URL: http://www.adacore.com/2009/05/26/astrium/

PARIS, NEW YORK and ISTANBUL, May 26, 2009 - DASIA 2009 - AdaCore, provider of the highest quality Ada tools and support, today announced that Astrium in the UK, a wholly owned subsidiary of EADS, dedicated to

providing civil and defense space systems and services, has selected the Ada programming language and AdaCore's GNAT Pro development environment for use on the new Sentinel-1 environmental monitoring satellite.

Sentinel-1 is the first of five families of satellites being developed for the Global Monitoring for Environment and Security (GMES) program. GMES, a joint initiative of the European Commission and the European Space Agency (ESA), is designed to support a sustainable European information network by monitoring, recording and analyzing environmental data and events around the globe. The Sentinel-1 project is scheduled to be completed in October 2011.

This instrument provides high-resolution satellite images by extending radar capabilities to penetrate forests and scrub to reach the ground, and registering any movements or changes on the Earth's surface within a resolution down to 5 meters. Astrium in the UK will use GNAT Pro to implement the Application Software for the SAR Electronics Sub-system which is used to control Sentinel-1's C-band Synthetic Aperture Radar (SAR). As part of the project, AdaCore will also port the GNAT Pro for LEON development environment to the Sensor Electronics Sub-system (SES) on-board computer, enabling compliance with the ESA's stringent ECSS-E-40B and ECSS-Q-80B space engineering and product assurance standards. The LEON2 processor was commissioned by ESA and designed specifically for use in satellite systems.

"Sentinel-1 is a large-scale project that promises to deliver superior environmental monitoring capabilities," said Paul Southwood, software technical engineer, Sentinel-1 Onboard Software project, Astrium (UK). "We selected Ada based on its reliability and reusability and its strong track record in space applications. We also wanted a development partner that could provide the knowledge, flexibility and responsiveness to meet our specific needs and tight timescales. Astrium has worked with AdaCore on a number of successful satellite and space projects, including the TerraSAR-X, which was the first German radar satellite for Earth observation. This experience combined with company's partnership approach and support capabilities made AdaCore the perfect choice for the Sentinel-1 project."

"AdaCore has a long and successful history in the space industry," said José Ruiz, AdaCore's team leader on the GNAT Pro Sentinel-1 port. "We are pleased that Astrium in the UK has chosen GNAT Pro as its language technology solution for Sentinel-1, and we look forward to providing a product that will fully meet their project's requirements."

About Astrium

Astrium, a wholly owned subsidiary of EADS, is dedicated to providing civil and defense space systems and services. In 2008, Astrium had revenues of €4.3 billion ($6.0 billion) and more than 15,000 employees in France, Germany, the United Kingdom, Spain and the Netherlands. Its three main areas of activity are Astrium Space Transportation for launchers and orbital infrastructure, Astrium Satellites for spacecraft and ground segment, and Astrium Services for the development and delivery of satellite services.

EADS is a global leader in aerospace, defense, and related services. In 2008, EADS generated revenues of €43.3 billion ($60.6 billion) and employed a workforce of more than 118,000.

www.astrium.eads.net

[…]

AdaCore — GNAT Pro used for C-130J software

From: AdaCore Press Center
Date: Monday June 1, 2009
Subject: Lockheed Martin Selects GNAT Pro for C-130J Software
URL: http://www.adacore.com/2009/06/01/c-130j/

SAN DIEGO, NEW YORK and PARIS, June 1, 2009 - Avionics USA - AdaCore, a leading supplier of Ada development tools and support services, today announced that Lockheed Martin Aeronautics, Marietta, Georgia, will be using GNAT Pro to develop the Flight Management System Interface Manager and Radio Control software on the C-130J Super Hercules aircraft. The specific product is GNAT Pro High-Integrity Edition for a PowerPC target running VxWorks 653, the time- and memory-partitioned real-time operating system from Wind River Systems.

The Lockheed Martin C-130J Super Hercules is an advanced tactical airlifter, designed for mission flexibility, combat delivery, air-to-air refueling, special operations, disaster relief, and humanitarian missions. Its range, power, performance, safety redundancy, reliability, and sophisticated avionics allow the aircraft to meet demanding mission requirements. With Rolls-Royce AE2100D3 engines and Dowty R391 six-bladed composite propellers, the Super Hercules can operate in hot climates, and handle short, high-elevation airstrips with maximum payload.

The Super Hercules transports 33% more payload, using half the crew, while burning less fuel and flying faster, farther and higher than its predecessors.

- Faster: The C-130J is faster, climbs more quickly, and offers 21% more speed. Time-to-climb is 50% better than earlier model C-130s.
- Higher: The C-130J flies higher, climbs over the weather and has a 40% greater cruising altitude than the C-130H.
- Farther: The C-130J flies much farther with less fuel, providing up to 40% greater range than the C-130H.

GNAT Pro is being used for the Block 7.0 software upgrade of the C-130J - the second cooperative Block Upgrade initiative that is a true international partnership, with the development costs shared among the participating nations, including the US government. This upgrade includes a new Flight Management System developed cooperatively between GE Aviation, Grand Rapids, Michigan, and Lockheed Martin Aeronautics in Marietta, Georgia. Since GE Aviation has used the GNAT Pro development environment on the 787 and C-130AMP upgrade programs, Lockheed Martin's selection of GNAT Pro will make it easier for the two companies to work together on the C-130J upgrade.

"AdaCore has been providing Lockheed-Martin with Ada development technology and support services for many years, and I look forward to continuing this relationship on the C-130J upgrade project," said Robert Dewar, President and CEO of AdaCore. "Avionics is a domain where errors can have catastrophic consequences, and I am pleased that Ada and GNAT Pro are being recognized as technologies that can provide the necessary safety and reliability."

In the United States, the C-130J is used by the U.S. Air Force, Air Force Reserve, Air National Guard, Marine Corps, United States Air Force Special Operations Forces, Mission Rescue Units and Coast Guard. International C-130J operators include the United Kingdom, Australia, Italy, Denmark and Norway, with Canada, India, Qatar and Iraq soon joining their ranks.

Indirect Information on Ada Usage

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

Job offer: [United Kingdom]: Avionics Developer (Ada 95, C)

[…]

I am looking for a developer with at least 3,5 years of experience in Ada 95 and C.

This role will be to join our client based team. This team performs activities in the domain of on-board software

development, data processing and avionics validation.

Candidates should also have demonstrable experience in Avionic & Functional Validation domains (or similar complex domain), as well as excellent understanding of Avionic and / or Functional Validation technical leads and Real-time HW-SW integration involving computer and buses (e.g. 1553B, Spacewire etc).

The candidates will support the validation of the avionic components software, define AOCS [Attitude and Orbit Control System —mp] test cases and execute test cases and analyse the test results.

The role will also involve the investigation and determination of the cause of test failures and production of test reports.

[…]

Job offer [United Kingdom]:

[…]

Experience of the full development and review lifecycle for safety/mission critical systems specified in DOORS, designed in Simulink and written in C, Ada 83 or Ada 95.

Software Engineers with experience in either low level testing of safety/mission critical systems using AdaTest or requirements based software integration testing in simulated environments.

Knowledge of DO-178B and/or MISRA would be beneficial, as would knowledge of Artisan and UML.

To apply for this position, candidates must be eligible to live and work in the UK and capable of gaining security clearance.

Job offer [United Kingdom]: Principal Engineer - Hardware in the Loop

My client develops some of the world's most advanced defence systems; they are looking for a Principle Engineer to be responsible for the full range of HWIL activities and be the subject matter expert across relevant projects.

This is a 'hands-on' role with many facets requiring an individual with a broad spectrum of skills and knowledge including: system modeling and design, software design, electronics design, as well as real-time computing and interfacing device drivers.

The primary tasks for this role will include:

- the design, development, implementation and maintenance of real-time simulation environments
- i.a.w. project Technical Requirements including the modification of design reference models
- production and testing of software to support the overall simulation including control, input/output interfaces and data recording/analysis tools
- integration and commissioning of hardware into the test environment
- verification and validation of the overall simulation including the target generation and presentation systems
- development of simulation environments from first principles, resolution of complex problems and flexibility in approach by adapting to changing requirements

Candidates for this role will be expected to have:

- A Degree or equivalent experience
- Proficiency in electronics, control systems, signal processing and mechanics
- Working knowledge of real time computing systems and architectures
- Previous experience in HWIL simulation (advantageous)
- Matlab/Simulink Experience
- Computer languages: C, C++, FORTRAN, Ada
- Requirements capture skills including interface definitions
- The ability to visualize and analyse complex systems

An ideal background for a suitable candidate would include one of the following:

- Missile/Weapon Systems
- Combat Systems
- Command & Control Systems
- Radar System Design
- Control Systems
- Military Communications Systems
- Data Link Systems

Job offer [United Kingdom]:

Essential skills and experience:

A proven track record in the design and development of safety-critical systems.

Essential technical skills and experience include:

- A BSc or higher in a numerate degree (e.g. Computer Science)
- Ada
- UML
- Requirement capture (DOORS)
- Database development
- Experience in using configuration management tools

[…]

A current UK security clearance would be an asset. It must be possible to clear those not already holding a clearance.

Desirable skills and experience include:

- Experience in avionics, transport and defence sectors
- Safety-critical related standards (e.g. DO-178B, EN 50128)
- Familiarity with C/C++, J2EE or .Net
- Familiarity with SysML
- Experience with Enterprise Architect and Artisan
- Systems Engineering

Job offer [United States]:

[…]

The successful candidate can expect to participate in the design, development and integration of fighter aircraft sensor systems for real-time, man-in-the-loop simulations.

This software models will be developed in either C++ and/or Ada, depending upon overall system application.

In general, a Bachelor Degree in Electrical Engineering is preferred for this position. However, other technical disciplines (Physics, Math, Aerospace, Computer Engineering, and Computer Science) will be considered along with relevant work experience for applicability to this position.

Prior use of C/C++ or Ada software languages is required as well as direct experience with real-time software systems and environments.

[…]

Qualifications:

C++ or Ada Software Development Experience

Software Modeling/simulation Experience

Real-time, embedded systems experience

[…]

Job offer [Spain]:

We are looking for a Computer Scientist, Software Engineer or Telecommunications Engineer with experience in Aerospace systems and good knowledge in C++.

Required:

- experience in applying aerospace system / software engineering processes and standards (especially ECSS)
- experience in UML, object-oriented development processes techniques and tools
- experience in high-integrity, embedded, real-time systems
- experience in C++
- high level of English

Desirable:

- experience in model-driven development
- experience in space transportation systems

- experience in integrated modular avionics
- experience in Ada
- French

Job offer [Spain]:

Technical Engineer or higher / MSc in Physics, Mathematics or Computer, Industrial, Telecommunications or Aerospace Engineering.

- Experience of 1/3 years in software engineering for safety-critical/real-time systems (railways, aerospace, defence, ATM)
- Excellent knowledge of Ada 83 / 95, C, C++ and UML
- Knowledge of development methodologies and software quality assurance is an advantage: CMMi, CENELEC, DO 178-B…

[Translated from Spanish —mp]

Job offer [Belgium]:

Context and objectives

- The project consists in participating in the software design and development, enhancement and testing of software systems used for co-ordination in Air Traffic Management (ATM) within Air Traffic Flow Management (ATFM) and between ATFM, Air Traffic Control, Air Space Management, Airports and Airlines.
- Our client collects, validates and corrects flight plans for all air traffic crossing European airspace, forwards them to ATC centers and aerodromes, matches this traffic demand against capacity, and, after collaborative decision making between flow managers and ATC, regulates traffic to avoid overloads by giving a minimal delay to non-exempted flights, with respect for equity between the operators.

Required profile

- Degree in Computer Sciences or Engineering.
- Ability to read, write and speak English is mandatory.
- Experience in Ada is mandatory. Ada 95 knowledge is a must.
- C++ knowledge is an advantage.
- Good experience in design and implementation of application software using OO techniques.
- Good knowledge of relational databases (e.g. Oracle).
- Experience in development tools like Unix scripting, Emacs, test tools, ClearCase is an advantage.
- Experience in HP-UX, LINUX is an advantage.
- Motivated in the development of new Air Traffic Management systems.
- Strong algorithmic knowledge.

Jobs requirements

- Being able to absorb large amounts of complex information. It is a necessary requirement to be able to maintain and to implement new requirements in a code base of roughly 1.5 MSLOC of Ada code.

[…]

Ada in Context

Miscellaneous math routines

From: johnscpg@googlemail.com
Date: Tue, 12 May 2009 09:38:49 -0700 PDT
Subject: ANN: miscellaneous Math routines, GPL'd
Newsgroups: comp.lang.ada

[…]

For those of you who like Ada with their numerics (is there anyone who doesn't?!) I have released a collection of math routines under the GPL license. Find them at:

http://web.am.qub.ac.uk/users/j.parker/miscellany

The full set is tarred in the file:

miscellany.10may09.tar.gz

in the directory given above.

Most are old classics I have found useful over the years (SVD, QR, LU, Runge-Kutta, FFT, Arbitrary precision floating point).

The random number generators are very new; the documentation should explain why I recommend them. The random number generators and the Arbitrary precision floating point are designed to make good use of the new 64-bit CPUs.

From: John B. Matthews <jmatthews@wright.edu>
Date: Tue, 12 May 2009 14:45:31 -0400
Subject: Re: ANN: miscellaneous Math routines, GPL'd
Newsgroups: comp.lang.ada

Excellent. You might like to see my experimental implementation of Generic_Roots, using the Durand-Kerner-Weierstrass method and distributed under the GNAT modified GPL:

http://home.roadrunner.com/~jbmatthews/misc/groots.html

[see also “Generic_Roots” in AUJ 29.4 pag.241 —mp]

From: johnscpg@googlemail.com
Date: Wed, 13 May 2009 02:39:50 -0700 PDT
Subject: Re: ANN: miscellaneous Math routines, GPL'd
Newsgroups: comp.lang.ada

Thanks for the pointer. Root finding BTW is one of those problems that cries out for extended precision floating pt. It’s rarely time-sensitive (rarely in inner loops), but very sensitive to precision…

Last time I did it, it was on Mathematica in extended precision…

I prefer Ada! If I did in Fortran I would always use REAL*16 (Quad-precision, 32 digit floats) but only one Fortran (intel ifort) supports this as far as I know (except IBM's Fortran on powerpc). If I ever find the time I'll hook up your root finder to my Extended Precision package.

[…]

From: Jerry Bauck <lanceboyle@qwest.net>
Date: Tue, 12 May 2009 14:02:56 -0700 PDT
Subject: Re: ANN: miscellaneous Math routines, GPL'd
Newsgroups: comp.lang.ada

Wow--numerical code that I can actually read! This looks awesome.

I suppose that it would not be hard to adapt the code to use the vector and matrix declarations in the Annex G.3 part of Ada 2005, e.g.

```
type Real_Vector is array
  (Integer range <>) of Real'Base;
type Real_Matrix is array
  (Integer range <>, Integer range <>)
  of Real'Base;
```

replacing the scattered declarations such as

```
type Data_Array is array (Array_Index)
  of Real;
type A_Matrix is array
  (R_Index, C_Index) of Real;
```

I really like the "standardized" way of these Ada 2005 declarations.

From: johnscpg@googlemail.com
Date: Wed, 13 May 2009 02:25:30 -0700 PDT
Subject: Re: ANN: miscellaneous Math routines, GPL'd
Newsgroups: comp.lang.ada

To instantiate the lin. alg. generics (in directory linearly) using the unconstrained arrays you typed above, declare a subtype m3:

```
subtype Index is Integer range 1..191;
type Real_Matrix is array
  (Integer range <>, Integer  range <>)
  of Real;
subtype m3 is Real_Matrix
  (Index, Index);
```

The generics can now be instantiated with m3.

Another interesting question is whether the generics (and the subprograms declared by the generic package) should be using unconstrained arrays:

```
generic
   type Real is digits <>;
   type Matrix is array
   (Integer range <>, Integer  range <>)
   of Real;
```

In the old days if I did that the program would run way slow.. seemed to put the compiler under much stress… maybe ok now but I am not going to be the one to prove it.

Here's another problem: Very common for an application to require transformation of an arbitrary diagonal block of a matrix (or any block) rather than the full matrix.

Unconstrained arrays don't provide any special benefit here…

you still have to copy the block to another matrix, or handle it the way I do in these routines (tell the routine to operate on a specified block only). Once you've gone to this trouble, the benefit of unconstrained arrays is small.

Another irritation: suppose user declares

```
V : Vector (1..10);
M : Matrix (0..9, 0..9);
```

or worse

```
M : Matrix (0..9, 2..11);
```

or worse still

```
M : Matrix (0..11, 2..11);
```

How do you handle it?

1. shift indices during iteration inside loops in the lin alg program.

2. slide arrays so indices coincide.

3. Raise constraint_error;

4. Assume they won't do it.

I do 3. (See for example procedure Choleski_Decompose at end of Disorderly-Random-Deviates.adb and near bottom of Disorderly-Random-Deviates.ads.). But I'm not sure what to do. (I was going ask here on comp.lang.ada, but got lazy.)

Generics with constrained arrays work ok for me here.

From: Gautier de Montmollin <gdemont@users.sourceforge.net>
Date: Wed, 13 May 2009 04:52:16 -0700 PDT
Subject: Re: ANN: miscellaneous Math routines, GPL'd
Newsgroups: comp.lang.ada

It's done here:

http://www.cs.umbc.edu/~squire/adaclass/gnatmath95 ,

generic_real_linear_equations.ads .

For using with Ada 2005's matrices, all you need is to replace package Generic_Real_Arrays by Ada.Numerics.Generic_Real_Arrays

From: johnscpg@googlemail.com
Date: Wed, 13 May 2009 06:43:11 -0700 PDT
Subject: Re: ANN: miscellaneous Math routines, GPL'd
Newsgroups: comp.lang.ada

Thanks Gautier! After a quick inspection, I notice a lot of method 1, (shifting indices inside loops), but sometimes the author copies the entire input array over to a new local array with the desired indices (usually 1..N).

Also raises an exception if there's a length mismatch.

gcc-ada for Solaris x86_64

From: Anonymous
Date: Fri, 27 Mar 2009 13:22:19 +0000 UTC
Subject: Build gcc-ada on/for Solaris x86_64?
Newsgroups: comp.lang.ada

I would like to be able to use GCC ada on my Solaris 10 Intel box. I did some preliminary checking and it doesn't look like a trivial undertaking.

I'm wondering if any of the participants in comp.lang.ada have done this or anything similar.

I have GCC 3.4.x running on the Solaris box from Sun's Companion CD but it was not built with Ada support.

I have 64 and 32 bit Linux boxes both running GCC 4.2.4 with Ada. Maybe it's simple (!) to build a cross target from this system?

I read somewhere that the GCC Ada version isn't necessarily the same as the GCC version. Is there a reason why I can't use 4.2.4 Ada on a GCC 3.4 base? Is gcc-ada at the 3.4 level worth having?

[…]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 27 Mar 2009 08:06:53 -0700 PDT
Subject: Re: Build gcc-ada on/for Solaris x86_64?
Newsgroups: comp.lang.ada

[…]

I suggest you install the package gcc3ada from www.blastwave.org; this is GCC 3.4.5 with Ada enabled. If you would like a more recent version, use gcc3ada to compile GCC 4.3.3 with Ada.

It would be nice if you would then contribute your packages back to blastwave.

PS. I recommend against package gcc4ada (GCC 4.0.2) because of bugs resulting from the introduction of Tree-SSA in GCC 4.0 (i.e. impedance mismatches between the Ada front-end and the GCC 4.0 back-end).

AuroraUX Project

From: AuroraUX Project
Date: Mon, 01 Jun 2009
Subject: Core Operating System for High Integrity Scientific Computing
URL: http://auroraux.blastwave.org/index.php/Main_Page

A few words about the project

AuroraUX is a Solaris-derived kernel- and user-land. The core of the project are its utilities written in Ada. When necessary, poorly implemented features get fixed or rewritten, as well.

Ada?

Yes, Ada. Ada was chosen because it encompasses every ideology that the core developers believe should exist in a system.

Ada has more compile time checks than you can shake a stick at and she keeps an eye on many things at runtime, too.

Code can generally be read by just about anyone with english language skills meaning that documentation can actually be written by a "normal" person.

The language was designed such that even it's syntax promotes integrity of software. But many thanks go to the powerful type system. These are only the tips of an iceberg and not necessarily the most important reasons.

Of course, some have shunned Ada saying it's a military language or it's an avionics language. Yes, and C is a text-processing language and C++ is for mobile phones.

And while we won't be able to escape the grips of C for obvious reasons, we can certainly make the world a happier one by throwing out (with discretion, for now) as much as we can and replacing it with an improved userland crafted with a language designed for real-time, embedded, safety-critical, reliable and maintainable systems by using our favorite lady, Ada.

AuroraUX is a noble, non-trivial goal, but with the determination and spirit of the existing developers, it will flourish-- though they would appreciate any help that is offered. :)

[…]

Project Constellations

Major components are meticulously documented.

- Hydra Custom package manager compatible with SVR4 packages
- Misc GCC Ada frontend GNAT GPL and tool-chain
- Nebula Installer featuring textual and graphical installers
- Singularity AuroraUX Kernel with userland written in Ada

- StarDust AuroraUX projects with complementary software from Blastwave, ISV and user projects
- SuperNova X.org with an desktop environment written in Ada
- Universe Common Tools and development items moved from Misc as needed

[verbatim from the home page of the project —mp]

Arguments for using Ada

From: Georg Maubach
Date: 19 May 2009 08:41:24 GMT
Subject: Arguments for using Ada
Newsgroups: comp.lang.ada

[…]

I am preparing a general suggestion for improvement into the group-wide ideas and innovation management system of my employer. For this I would like to back up me suggestion with some articles about the advantages of using Ada.

I have so far:

Proof that a software development project using Ada takes less - instead of more time - than expected: "The Return of Ada"

http://gcn.com/Articles/2008/04/11/The-return-of-Ada.aspx?Page=1

Proof that complex systems within the logistics and transportation industry work more reliable and can be better maintained using Ada than other languages: "European Train Control System"

http://www.adaic.com/atwork/euro.html

List of software systems based on Ada that work without defects and are therefore never heard of

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

http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/success/success.html

Ada is tought at German universities, programmers could be recruited from there

http://www.ada-deutschland.de/ful/index.html

Proof of the activities of the Ada user community:

- FOSDEM 2009 - Ada Developers Conference Room

http://www.cs.kuleuven.be/~dirk/ ada-belgium/events/09/090207-fosdem.html

- Ada Europe Conference 2009

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

- ACM SigAda Conference 2009

http://www.sigada.org/conf/sigada2009/

- Discussion Group

comp.lang.ada

Tools and Utilities can be bought from many manufacturers, supplier change is possible if needed, no single supplier dependency

http://www.ibm.com/software/awdtools/developer/ada/, http://www.eds.com/, http://www.adacore.com, http://www.aonix.com, http://www.ddci.com, http://www.ghs.com

Training courses are available from different suppliers

http://www.ddci.com/, http://www.abssw.com, http://www.classwide.com

What is missing are articles relating to the following:

- There are enough Ada programmers available.
- Ada can be used with more success than other languages. There was a software development project at a university which asked a first group of students to use C (not successful) and asked a second group of students using Ada (successful).
- Links with content (e.g. research) that proofs that development with Ada results in less defects than programs in other languages
- Links with content (e.g. research) that proofs that development with Ada results in better maintainable software

Can you help me with links and information on the aspects still missing?

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Tue, 19 May 2009 02:13:39 -0700 PDT
Subject: Re: Arguments for using Ada
Newsgroups: comp.lang.ada

> […] There was a software development project at a university which asked a first group of students to use C (not successful) and asked a second group of students using Ada (successful).

John McCormick's Model Railroad:

http://www.adaic.org/atwork/trains.html

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 19 May 2009 02:18:04 -0700 PDT
Subject: Re: Arguments for using Ada
Newsgroups: comp.lang.ada

> Tools and Utilities can be bought from many manufacturers, supplier change is possible if needed, no single supplier dependency […]

I don't think this is relevant to most pointy-haired bosses.

They're content to become a captive customer of Microsoft, Sun (Java) and Intel (processors). Instead, I think the relevant idea is that they can choose one reliable supplier that provides support in the long term, and happily become captive.

> Training courses are available from different suppliers […]

Also www.adalog.fr.

> What is missing are articles relating to the following:

>

> - There are enough Ada programmers available.

At Eurocontrol we never have a problem recruiting Ada programmers, or people willing to convert to Ada. But pointy-haired bosses like to point out "the lack of Ada programmers" when they run out of excuses for poor engineering decisions.

If a programmer can learn Java or C#, they can learn Ada.

[…]

> - Links with content (e.g. research) that proofs that development with Ada results in less defects than programs in other languages

http://www.adaic.com/whyada/ada-vs-c/cada_art.html

> - Links with content (e.g. research) that proofs that development with Ada results in better maintainable software

http://www.adaic.com/whyada/ada-vs-c/cada_art.html

> Can you help me with links and information on the aspects still missing?

http://www.adaic.com/whyada/

I think the biggest hurdle to overcome is the lemming mentality; even if you point out that a pointy-haired boss is thinking like a lemming, they are still happy about that because they want their programmers to be interchangeable (i.e. disposable). Some do not even understand the value of good education and tools for good software engineering.

Also, before you convince pointy-haired bosses, you need to win the hearts and minds of developers. So, I would suggest you discuss your idea with some developers first, and not try to have managers force Ada on them. You will be better able to convince managers if several top developers back your suggestions.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 19 May 2009 02:38:16 -0700 PDT
Subject: Re: Arguments for using Ada
Newsgroups: comp.lang.ada

There is statistical proof that the number of Ada programmers is increasing:

http://qa.debian.org/popcon-graph.php?packages=gnat+gnat-4.1+gnat-4.3&show_installed=on&want_legend=on&want_ticks=on&from_date=&to_date=

&hlght_date=&date_fmt=%25Y-%25m&beenhere=1

From: Roderick Chapman <roderick.chapman@googlemail.com>
Date: Tue, 19 May 2009 04:28:38 -0700 PDT
Subject: Re: Arguments for using Ada
Newsgroups: comp.lang.ada

> - Links with content (e.g. research) that proofs that development with Ada results in less defects than programs in other languages

I would add "The existence of SPARK" to your list of things that are special and/or notable about Ada.

For data on defect rates, see the various publications regarding SPARK usage on www.sparkada.com

Also see Andy German's paper from CrossTalk (google for "German Crosstalk QinetiQ").

See the Tokeneer system from www.adacore.com/tokeneer

From: Peter Hermann
Date: Tue, 19 May 2009 12:29:39 +0000 UTC
Subject: Re: Arguments for using Ada
Newsgroups: comp.lang.ada

"some" more links:

http://www.ihr.uni-stuttgart.de/forschung/ada/resources_on_ada/

From: Mike Silva <snarflemike@yahoo.com>
Date: Thu, 21 May 2009 07:23:23 -0700 PDT
Subject: Re: Arguments for using Ada
Newsgroups: comp.lang.ada

I've always been impressed by the finding that some Ada code certified to DO-178B had only one tenth the residual error rate of some C code also certified to DO-178B.

Maybe the Praxis folk can help you document that finding.

In addition, you should peruse all their online publications.

Here's the reference to the one-tenth residual error figure:

http://www.praxis-his.com/sparkada/pdfs/spark_c130j.pdf

From: Roderick Chapman <roderick.chapman@googlemail.com>
Date: Thu, 21 May 2009 11:33:10 -0700 PDT
Subject: Re: Arguments for using Ada
Newsgroups: comp.lang.ada

[...]

> Maybe the Praxis folk can help you document that finding.

It's in Andy German's paper from CrossTalk - see my posting above.

Advantages of Ada over C for embedded systems programming

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 7 Mar 2009 14:16:28 +0100
Subject: Re: C vs. ada for embeded system
Newsgroups: comp.lang.ada

> why ada is better of "C" language for embedded system?

That depends on many factors. What worked for us was:

1. Portability. We develop and test under Windows. The target platform is used only incidentally. We use same code for both, no single line of preprocessor;

2. Language standard;

3. Tasking support (yes, we needed tasks);

4. OOPL;

5. Excellent support (AdaCore).

Disadvantages:

1. Damn difficult to get a compiler;

2. Tool chain (IDE, debugger etc) for Ada is not that good. Well not bad when compared with Workbench/Eclipse, but still no match to Visual Studio;

3. Initial costs are much higher than for C.

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Sat, 7 Mar 2009 10:12:55 -0800 PST
Subject: Re: C vs. ada for embeded system
Newsgroups: comp.lang.ada

[…]

For me the features which aid embedded systems in particular are:

1. Built-in real-time clock

2. Built-in fixed point numbers

3. Portability

4. Less buggy*

* that's taking an average bunch of engineers who know their chosen language to a decent level. You can get dummies writing cr*p Ada and super-programmers turning out bug-free C. Which matches your current team? Which matches your future team?

The features that are in the 'pro' column for Ada that aren't specific to embedded systems are:

1. Ranged elementary types (i.e. int and float with ranges that run-time checks)

2. Built-in tasking

3. No pre-processor

4. Proper arrays (=> proper Strings)

5. Built-in container libraries

6. Exceptions

7. Packages (i.e. proper modules - not faked up with 'guard macros' => no unintentional dependencies via '#include')

8. Named parameter association

[…]

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Sat, 07 Mar 2009 18:29:08 GMT
Subject: Re: C vs. ada for embeded system
Newsgroups: comp.lang.ada

[…]

Where we have hard data, they show that Ada results in delivery at half the cost and with 1/4 the errors of C, and the errors cost 1/10 as much to correct.

(Of course, that's not specific to embedded systems.)

From: Per Sandberg <per.sandberg@bredband.net>
Date: Sat, 07 Mar 2009 20:54:18 +0100
Subject: Re: C vs. ada for embeded system
Newsgroups: comp.lang.ada

Well, would say that that depends entirely where you think its worth spend your time. 100 hours upfront designing and fighting the compiler or 500 hours after delivery fighting with the debugger.

And in my experience that counts for all kinds of systems.

From: Anonymous
Date: Sat, 07 Mar 2009 23:26:39 GMT
Subject: Re: C vs. ada for embeded system
Newsgroups: comp.lang.ada

Ada is a more perfect portable language while C has become a damaged language by the use of macros and conditional statements. Which initially allowed C and other software written in C to become portable has now branched into a nightmare for programmers to maintain the older software.

Ada has no language defined macros or conditional statements only a number of packages that need to be rewritten for each platform and/or OS used. And the specific machine and platform dependent code of these packages are transparent to the programmers and their software. Giving a better and more portable language to use.

Plus, the built-in features like tasking and safety makes Ada a winner. Of course, the closer you get to perfection for a language the higher the cost.

[…]

From: John McCormick <mccormick@cs.uni.edu>
Date: Mon, 9 Mar 2009 09:02:32 -0700 PDT
Subject: Re: C vs. ada for embeded system
Newsgroups: comp.lang.ada

You can see the results of completion rates for a 10-15K line project done in both C and Ada in my Real-Time Embedded Systems class over the past 20 years. The class runs for one semester (15

weeks). The bottom line is that NO student team ever completed the minimal project requirements in C. Over 80% of the teams working in Ada completed the project. My latest publications on the matter are:

"We've been working on the railroad: a laboratory for real-time embedded systems", McCormick, J. W., SIGCSE Bulletin, Volume 37, Number 1, Feb. 2005, Page(s): 530-534.

"Instrumentation education through model railroading", McCormick, J.W., Instrumentation & Measurement Magazine, IEEE, Volume 9, Issue 5, Oct. 2006 Page(s): 40 - 45

"Model Railroading and Computer Fundamentals", McCormick, J.W., Journal of Computer Science Education, Volume 17, Number 2, June 2007, Page(s): 129 - 139

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Tue, 10 Mar 2009 18:33:24 +0100
Subject: Re: C vs. ada for embeded system
Newsgroups: comp.lang.ada

[…]

Interesting - where the students allowed to choose the language or was it forced upon them?

Do the students know before hand of the dire C results?

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Wed, 11 Mar 2009 03:14:47 -0700 PDT
Subject: Re: C vs. ada for embeded system
Newsgroups: comp.lang.ada

[…] you can find the article here:

http://www.adaic.com/atwork/trains.html

It's a very interesting reading and answers all your questions.

From: John McCormick <mccormick@cs.uni.edu>
Date: Wed, 11 Mar 2009 07:07:01 -0700 PDT
Subject: Re: C vs. ada for embeded system
Newsgroups: comp.lang.ada

> Interesting - where the students allowed to choose the language or was it forced upon them?

Forced :) Christoph pointed out an older article that is easy to access. We used C the first 7 years leading to a very frustrated professor. Even when I gave them over 50% of the C code, they still failed. Two attempts were made by Ada graduates to write the systems in C++ and Real-Time Java. Neither succeeded.

I have a video of the laboratory at

http://www.cs.uni.edu/~mccormic/RealTime/

You can also see a user's manual that describes the system the students implement.

A comparison between Ada and Eiffel

From: Tomek Walkuski <tomek.walkuski@gmail.com>
Date: Sun, 24 May 2009 01:39:14 -0700 PDT
Subject: Ada vs Eiffel - Ada programmer approach
Newsgroups: comp.lang.ada

[…]

I do not want to start another flame war which language is better.

I think that Ada and Eiffel target same field and I want to ask you, Ada programmers, about:

- what, in your opinion, is better in Ada, in contrast to Eiffel?
- what, in your opinion, is worse in Ada, in contrast to Eiffel?

Please, share your thoughts if you have an experience in both languages.

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Sun, 24 May 2009 14:17:51 +0200
Subject: Re: Ada vs Eiffel - Ada programmer approach
Newsgroups: comp.lang.ada

[…]

- Eiffel has garbage collection by default. Good for programs that can afford it.
- Eiffel offers a little less to support systems/hardware programming directly. For example, the base type system is frozen, which means that you cannot have basic types of the range 0 .. 100 kind, the advice being to use DbC assertions instead. For bit operations we can use features from class NATURAL_32…
- There is class POINTER/ POINTER_REF; it may be necessary to use "extern", i.e. C, for hardware address things… Not sure about the latter, though. Ada's loop is more flexible, which I guess matters when a program is to run in 64kB or less, or when I want tight control over inner loops.
- Eiffel has, or used to have, a "standardized" separate cluster language for namespace things and also configuration. (Currently being, uh, reworked?) I like the configuration part because it removes vendor specific language without impeding vendors I should think. You can't see the "namespaces" in the source proper. Good or bad? Advice is: Use EiffelStudio.
- Eiffel's concurrency features are basically experimental. (Side note: the Redmond amoeba has recently embraced the word "task".) While Eiffel's "separate" and SCOOP are defined, class THREAD is mentioned a lot… The SmartEiffel language is said not to be thread safe at this time.
- Ada's separate specification is only possible using deferred classes in Eiffel. You can then use the cluster language to rename this or that class as a "body" class, implementing any deferred feature. Using deferred this way seems like a misuse to me.
- Eiffel has full multiple inheritance. Not sure this is a big deal, in practice, compared to mix-in generics, but it is being used to add characteristics such as HASHABLE or INDEXABLE. Seems somewhat less "flexible" than Ada generic actuals that can be supplied from just about anywhere, including from local scopes. (Not sure there are no Eiffel tricks here, using [anonymous] agents, though).
- Eiffel's Design by Contract is only beginning to be present in Ada. (Not mentioning SPARK.) Still praying…
- Eiffel is flat (anonymous agents being an exception of sorts…), Ada allows nesting, including local types.

I really missed the explicit packaging structure of Ada when writing the same (sequential) program in both Eiffel and Ada.

From: Pascal Obry <pascal@obry.net>
Date: Sun, 24 May 2009 18:31:24 +0200
Subject: Re: Ada vs Eiffel - Ada programmer approach
Newsgroups: comp.lang.ada

[…]

> - what, in your opinion, is better in Ada, in contrast to Eiffel?

- a strong standard (Eiffel community is sadly split these days).
- string type system
- tasking
- distributed annex
- support for interfacing with hardware (e.g. representation clauses).
- not everything is a class (composability of Ada, packages, tagged types, child packages...)
- more components and binding to common libraries
- bigger and more active community

> - what, in your opinion, is worse in Ada, in contrast to Eiffel?

- memory management (not GC - I know that there is nothing in the standard that forbid it - and this would be certainly really handful for some domain)
- multiple inheritance
- DbC with pre-condition/post-condition/invariant

[…]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>

Date: Tue, 26 May 2009 06:37:49 -0700 PDT
Subject: Re: Ada vs Eiffel - Ada programmer approach
Newsgroups: comp.lang.ada

[…]

> - memory management (not GC - I know that there is nothing in the standard that forbid it - and this would be certainly really handful for some domain)

I agree that it would be nice if some Ada compilers would provide an optional garbage collector. One exists as a third-party add-on to GNAT but I've never used it and it's not part of AdaCore's or the FSF's distributions.

In fact, I would like it if the garbage collector could optionally log all deallocations while the program runs, so I can find memory leaks, correct them, and remove the garbage collector later.

> - multiple inheritance

I'm not sure this is a good thing. In fact, I'm not sure simple inheritance is always a good thing, either. I tend to prefer composition and generics.

> - DbC with pre-condition/post-condition/invariant

When I read Bertrand Meyer's "Object-Oriented Software Construction", I too thought that DbC was a brilliant idea. Now I'm less convinced. I see two major drawbacks to DbC:

- pre/post conditions and invariants involve run-time checks most of the time (if not all the time). They slow the program down if enabled, or become useless when disabled for performance. I like static checking much better; Ada provides a lot of that out of the box (much more than Eiffel) and SPARK goes way beyond even that.
- in most of the examples I saw in the literature, only very simple subprograms would have a contract and the contract would mostly repeat the body of the subprogram. This redundancy is counter-productive. For more complex subprograms, it can be very difficult to write pre- and post-conditions and invariants; Ada's pragma Assert provides what I need in these (rare) cases because I can put such pragmas in the middle of a subprogram, for example.

From: Tim Rowe <spamtrap@tgrowe.plus.net>
Date: Tue, 26 May 2009 16:07:06 +0100
Subject: Re: Ada vs Eiffel - Ada programmer approach
Newsgroups: comp.lang.ada

[…]

I still think it's a brilliant idea, but suffers from *Design* by Contract getting confused with *Programming* by Contract; not least because "Design by Contract" is a trademark, so people needed a different name to refer to the approach, but I'm not sure Bertrand Meyer always kept the distinction clear either.

Both of the problems you describe are programming issues, not design issues. There's nothing about the *design* process that mandates any run-time checks or that requires you to have pre- and post-conditions expressed in the target language. That's a matter for coding standards. To me, Design by Contract simply means working out and documenting in advance under what circumstances a section of code can legitimately be entered, and, if that is satisfied, what we can guarantee on exit. Learning to do that rigorously and diligently has certainly improved my own design.

We had that before Meyer coined the term "Design by Contract" of course. We used to call it "specification".

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 26 May 2009 16:51:35 +0200
Subject: Re: Ada vs Eiffel - Ada programmer approach
Newsgroups: comp.lang.ada

[…]

> - in most of the examples I saw in the literature, only very simple subprograms would have a contract and the contract would mostly repeat the body of the subprogram. This redundancy is counter-productive.

That is when the contract is thought to be an equivalent of correctness proof. But it is not. A contract is much weaker, so that a program which fulfills its contract is not necessarily a correct program. Weaker contracts have an advantage to remain statically checkable for complex programs, leaving proofs of correctness aside.

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Tue, 26 May 2009 18:37:29 +0200
Subject: Re: Ada vs Eiffel - Ada programmer approach
Newsgroups: comp.lang.ada

> […] This redundancy is counter-productive.

I thought that too, but I don't consider redundancy counter-productive any more; it makes me think twice. In fact, trying to find a post-condition - which must be consistent with the module invariant, for which there isn't any support in Ada yet - has forced me to simplify program structure. Isn't that a good thing?

> For more complex subprograms, it can be very difficult to write pre- and post-conditions and invariants; Ada's pragma Assert provides what I need in these (rare) cases because I can put such pragmas in the middle of a subprogram, for example.

Eiffel has "check" and "debug" which are more specific than pragma Assert.

> Both of the problems you describe are programming issues, not design issues. There's nothing about the *design* process that mandates any run-time checks or that requires you to have pre- and post-conditions expressed in the target language. That's a matter for coding standards. To me, Design by Contract simply means working out and documenting in advance under what circumstances a section of code can legitimately be entered, and, if that is satisfied, what we can guarantee on exit.

This "section of code" bit is important; loop invariants and variants are part of Eiffel DbC.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 26 May 2009 19:39:39 +0200
Subject: Re: Ada vs Eiffel - Ada programmer approach
Newsgroups: comp.lang.ada

[…]

Whatever you check or assert at run-time that is not a contract. The effect of checking is a behavior. Contract check cannot be a behavior. That is the inconsistency of Eiffel's approach, as well as one of Ada's pragmas if they have any run-time effects.

Ada's static typing system and SPARK do it right.

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Tue, 26 May 2009 19:59:26 +0200
Subject: Re: Ada vs Eiffel - Ada programmer approach
Newsgroups: comp.lang.ada

SPARK imposes limitations that are not present when employing DbC. SPARK cannot replace DbC, or improve it, and vice versa, basically because DbC (not used as static assertions only) and SPARK are largely incommensurable.

This means that DbC and SPARK are both right: You will --# hide certain uses of Ada constructs in SPARK, for example. Likewise, you will not check all assertions in some situations when using Eiffel DbC.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 26 May 2009 23:28:47 +0200
Subject: Re: Ada vs Eiffel - Ada programmer approach
Newsgroups: comp.lang.ada

[…]

> SPARK imposes some very strong constraints on the developer (for good

> reasons) to build and *prove* something right.

Yes. The amount of checks and so the limitations imposed by them depends on the word "something" you used above. You can require to prove less or more, but never all.

Many programmers are already satisfied with much less than SPARK offers, e.g. with strong static typing, which gives a proof of no type errors. But, continuing this example, dynamic typing gives no such proof. Therefore it is either not strong (most of dynamically typed language are in fact weakly typed) or else the dynamic type checks (like dispatch in Ada) are not error checks, but merely correct contracted behavior.

In my view run-time checks as a form of contract enforcement is a bad practice, which makes the programmer believe in safety that does not exist.

Instead of that he should consider his design to define the behavior for the cases where the "contract" can be violated, making a new contract that is never violated and thus requires no checks.

On the platform independence of Ada

From: Patrick Gunia <patrick.gunia@googlemail.com>
Date: Wed, 11 Mar 2009 07:04:02 -0700 PDT
Subject: Ada Platform Independence
Newsgroups: comp.lang.ada

I'm currently working on an analysis of an Ada system focusing on the current state of platform independence. As my experience with the porting of Ada software to different operating system is limited and close to zero, I'd like to ask which concepts of the language might cause problems. I don't mean aspects like including platform-dependent libraries or calling system functions from Ada code. I know that this will lead to portability problems. I'm more interested in problems with the Ada language itself.
For example use of representation clauses or I/O statements.

I've been searching a while to find a listing of parts of the Ada standard which might cause problems, but didn't succeed.

[…]

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 11 Mar 2009 16:37:44 +0100
Subject: Re: Ada Platform Independence
Newsgroups: comp.lang.ada

[…]

Ada allows writing portable programs, however it does not guarantee in itself that programs will be portable. The language purposely allowed to depend on the peculiarities of the target. Care is still needed to achieve portable code.

Common difficulties include:

- Relying on the characteristics of predefined types (Integer and Duration)
- Differences on the implementation of Address. Some compilers (mainly Ada83) defined Address as an integer type, while current common practice is to make it private, which causes problems if people are doing (uncontrolled) address arithmetic
- Representation clauses. Compilers vary in their support of representation clauses, sometimes for good reasons: some representation clauses that are acceptable on some targets would lead to unreasonable code on a different hardware. Representation clauses may also depend on predefined types; f.e., if you have a record field of type Duration, you may have problems when moving a program from an implementation where Duration is 32 bits to one where Duration is 64 bits
- Outrageously wrong code written by people who write "C-in-Ada", with lots of unchecked conversions between pointers and addresses.

In practice, most portability problems are rooted in insufficient training of the people who wrote the code initially.

From: Patrick Gunia <patrick.gunia@googlemail.com>
Date: Wed, 11 Mar 2009 10:00:18 -0700 PDT
Subject: Re: Ada Platform Independence
Newsgroups: comp.lang.ada

[…]

> - Differences on the implementation of Address. Some compilers (mainly Ada83) defined Address as an integer type, while current common practice is to make it private, which causes problems if people are doing (uncontrolled) address arithmetic

I also thought of this aspect. I have the big advantage that the software is only executed on a very limited range of hardware configurations. Using the same compiler implementation for different platforms (for example GNAT) would solve this problem?

[…]

> - Outrageously wrong code written by people who write "C-in-Ada", with lots of unchecked conversions between pointers and addresses.

The code uses unchecked_conversions to call imported C-functions. I don't think that I can avoid this problem. Though the software is always executed on 32-bit systems, thus this should also work out, or am I getting something terribly wrong here?

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 12 Mar 2009 10:44:16 +0100
Subject: Re: Ada Platform Independence
Newsgroups: comp.lang.ada

> […] Using the same compiler implementation for different plattforms (for example GNAT) would solve this problem?

As far as I know, GNAT (and presumably most Ada 95 compilers, since it is implementation advice) define Address as private.

[…]

Unchecked_Conversion was commonly used in Ada83. Using pragma Import and Interfaces.C is preferred since Ada95. I would suggest migrating the code to use these features.

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Wed, 11 Mar 2009 10:33:49 -0700 PDT
Subject: Re: Ada Platform Independence
Newsgroups: comp.lang.ada

[…]

I've ported Ada code from 68k to i386 from no-OS to VxWorks to Win32 and in each case the biggest hurdle has been 'how good is the original code'.

Assumptions on the sizes and ranges, esp. of the predefined types (especially Integer), can cause problems.

Fixed-point numbers are different between revisions of the language (e.g. Ada83 vs. Ada95 vs. Ada2005) but should be portable enough between different ports using the same language. A solution is to providing your definition not dependent on a particular underlying representation being used.

The 'favourite' gotcha is usually using non-standard additions to standard packages. E.g. XD-Ada extended the contents of package 'System' to include 8/16/32-bit signed/unsigned integer. Easy to deal with - just define a package called 'XDAda_System' and defines subtypes of the types in Interfaces in it. Then do a global find/replace for all occurrences of 'with System' (and 'use'!) and you're 90% there.

I ported 45kSLOC from no-OS XD-Ada to GNAT GPL 2008 Windows/Intel in an hour the other week (where 'ported' means getting a clean compilation / link and not actually running it).

Interrupt Handling and Timing Events

From: Reto Buerki <reet@codelabs.ch>
Date: Fri, 15 May 2009 18:26:12 +0200
Subject: Interrupt handler and Ada.Real_Time.Timing_Events
Newsgroups: comp.lang.ada

I hit a rather strange issue today mixing signal/interrupt handling with Ada.Real_Time.Timing_Events. We have a real life application where we use timing events but we also need a signal

handler to catch signals from the environment (SIGTERM etc.).

I wrote a small reproducer to illustrate the problem. The following protected object is used as an interrupt handler, which can be attached to a specific interrupt/signal:

```
with Ada.Interrupts;
package Handlers is
  protected type Signal_Handler
    (Signal : Ada.Interrupts.Interrupt_ID)
  is
    pragma Interrupt_Priority;
    entry Wait;
  private
    procedure Handle_Signal;
    pragma Attach_Handler
      (Handle_Signal, Signal);
    Occured : Boolean := False;
  end Signal_Handler;
end Handlers;

package body Handlers is
  protected body Signal_Handler is
    procedure Handle_Signal is
    begin
      Occured := True;
    end Handle_Signal;
    entry Wait when Occured is
    begin
      if Wait'Count = 0 then
        Occured := False;
      end if;
    end Wait;
  end Signal_Handler;

end Handlers;
```

The handler is used like this:

```
with Ada.Text_IO;
with Ada.Interrupts.Names;
-- Uncommenting the next line breaks
-- interrupt handler
-- with Ada.Real_Time.Timing_Events;

with Handlers;
procedure Interrupt_Problem is
  use Ada.Interrupts;
  Handler : Handlers.Signal_Handler
      (Signal => Names.SIGTERM);
begin
  if Is_Attached (Interrupt =>
      Names.SIGTERM) then
    Ada.Text_IO.Put_Line ("Attached
                handler to SIGTERM");
  else
    Ada.Text_IO.Put_Line ("Could not
                attach to SIGTERM!");
    return;
  end if;
  Handler.Wait;
  Ada.Text_IO.Put_Line ("Interrupt
                received…");
end Interrupt_Problem;
```

As expected, when sending SIGTERM to the running 'Interrupt_Problem' process "Interrupt received…" is displayed. So far so good.

As commented in the source code, as soon as the Ada.Real_Time.Timing_Events package is with'ed, this mechanism breaks.

The signal handler is not invoked any more when I send a SIGTERM signal to a running 'Interrupt_Problem' process, it just terminates without triggering the Handler.Wait.

What could be the cause for this behavior? Is there a problem with this code?

[…]

From: Adam Beneschan <adam@irvine.com>
Date: Fri, 15 May 2009 09:54:17 -0700 PDT
Subject: Re: Interrupt handler and Ada.Real_Time.Timing_Events
Newsgroups: comp.lang.ada

[…]

My guess would be that when Ada.Real_Time.Timing_Events is with'ed, this causes elaboration code for the Timing_Events package to be executed (before Interrupt_Problem), and there must be something that this elaboration does that interferes with the Attach_Handler mechanism. I can't find anything in the language definition of Timing_Events that would cause this, so it must be a problem particular to your Ada compiler implementation, and you should contact the vendor, or at least let us know what compiler you're using so that others who may have some knowledge of that particular compiler might be able to help.

[…]

From: Reto Buerki <reet@codelabs.ch>
Date: Sat, 16 May 2009 01:24:04 +0200
Subject: Re: Interrupt handler and Ada.Real_Time.Timing_Events
Newsgroups: comp.lang.ada

[…] I'm using FSF GNAT 4.3.2 on Debian Lenny. […]

From: sjw <simon.j.wright@mac.com>
Date: Fri, 15 May 2009 23:28:47 -0700 PDT
Subject: Re: Interrupt handler and Ada.Real_Time.Timing_Events
Newsgroups: comp.lang.ada

On Mac OS X/GCC 4.3.3, the program as written outputs "raised PROGRAM_ERROR : Interrupt 15 is reserved".

Changed to SIGUSR1: now runs as designed.

Sending SIGTERM is ignored (this seems odd).

Uncomment Timing_Events: doesn't report anything, but ps shows "User defined signal 1 ./interrupt_problem"

and when I run ps again the process has gone.

This is all deep stuff (and apparently OS-dependent)!

[…]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 15 May 2009 09:56:54 -0700 PDT
Subject: Re: Interrupt handler and Ada.Real_Time.Timing_Events
Newsgroups: comp.lang.ada

[…]

Ada.Real_Time.Timing_Events's elaboration block creates a task and promotes it to an outer level (i.e. it is no longer dependent on a master).

The only way to terminate this task is by sending it SIGTERM, so the task attaches another signal handler to SIGTERM before yours.

That handler catches the signal and does not propagate it to any other handler.

See System.Task_Primitives. Operations.Initialize.

I'm afraid there is no way out :) maybe you can use another signal in your task?

From: Yannick Duchêne <yannick_duchene@yahoo.fr>
Newsgroups: comp.lang.ada

Subject: Re: Interrupt handler and Ada.Real_Time.Timing_Events
Date: Fri, 15 May 2009 16:24:36 -0700 PDT
[…]

Do you know why it is not propagated ?

SIGTERM is supposed to be intended to the whole of an application, not only to a part of it.

From: Reto Buerki <reet@codelabs.ch>
Date: Sat, 16 May 2009 02:20:54 +0200
Subject: Re: Interrupt handler and Ada.Real_Time.Timing_Events
Newsgroups: comp.lang.ada

I tried attaching the handler to various signals. As soon as the timer task is started in the Ada.Real_Time.Timing_Events elaboration block, my own handler is not triggered any more. This seems odd.

We are using Ada.Real_Time.Timing_Events to implement an event-driven architecture in our application. The Timing_Event type seemed perfect for this.

Nevertheless, the application should still be able to react to signals it may receive

from the operating system. Is it really Timing_Events XOR interrupt handling?

From: Jeffrey R. Carter <spam.jrcarter.not@nospam.acm.org>
Date: Sat, 16 May 2009 00:38:54 GMT
Subject: Re: Interrupt handler and Ada.Real_Time.Timing_Events
Newsgroups: comp.lang.ada

[…]

Timing_Events are certainly suited for this. However, they were added in the most recent revision of the language. Event-driven systems were implemented in Ada long before that revision. Should you be unable to get your application to work with Timing_Events, you can use the old-fashioned way to achieve the same thing. This is done by having tasks that execute delay statements and then call the appropriate protected operations.

Effectively, a Timing_Event object is shorthand for such a task.

From: Reto Buerki <reet@codelabs.ch>
Date: Fri, 29 May 2009 17:59:46 +0200
Subject: Re: Interrupt handler and Ada.Real_Time.Timing_Events
Newsgroups: comp.lang.ada

Thanks for your answer. Re-implementing the Timing_Events functionality seems to be the only possible solution for the moment.

I sent a bug report about this issue to report@adacore.com and added it to the GCC bug database (bug #40285).

On the overhead of exception handling

From: Peter C. Chapin <pcc482719@gmail.com>
Date: 24 Apr 2009 11:52:27 GMT
Subject: Exception handling overhead?
Newsgroups: comp.lang.ada

[…] I'm creating a type intended to represent lines of editable text. It will provide some specialized services and is not intended to be a general purpose string type. At the moment I'm wrapping the type Ada.Strings.Unbounded.Unbounded_String (I'm really using Wide_Strings, but I don't think that's important right now).

I may want to change that underlying implementation in the future so I don't want to directly expose my current choice to my clients.

Thus I'm providing some subprograms that do some minor things and then just forward to subprograms in Ada.Strings.Unbounded.

My question is about exceptions. To avoid exposing my implementation choice and to give me more control in the long run I've defined my own exception for (as an example) an out of bounds access:

```
Bad_Index : exception;
```

Naturally I want to raise this exception if the client tries to access a character not in my buffer. This leads to two possibilities.

```
function Element (B : Buffer;
                  Index : Positive)
   return Character is
begin
 if Index >
       Ada.Strings.Unbounded.Length
       (B.Internal_String) then
   raise Bad_Index;
 end if;
 return Ada.Strings.
        Unbounded.Element
        (B.Internal_String, Index);
end Element;
```

This entails two checks on the index value: the one I'm doing and the one being done inside Ada.Strings.Unbounded.Element.

Another approach is:

```
function Element(B : Buffer;
                 Index : Positive)
   return Character is
begin
 return Ada.Strings.
        Unbounded.Element
        (B.Internal_String, Index);
exception
 when Ada.Strings.Index_Error =>
      raise Bad_Index;
end Element;
```

This lets the check exist in only one place but it invokes the machinery of exception handling.

Now I know that in C++ there are implementation methods that allow zero execution time overhead in the case when an exception is not thrown (at the expense of increasing executable size and making thrown exceptions slower). Compilers are not required to use such methods, of course, but I know of at least one that does… or that can if the right options are selected.

I'm wondering about how this might work in Ada. Does Ada admit such implementation methods? If so, do compilers commonly use them? From a raw performance point of view which of the two approaches above would be "better?"

For the record… I understand that in my application it probably doesn't matter much either way. I also understand that the answer is likely to be very compiler specific. This is mostly a question of academic interest.

[…]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 24 Apr 2009 05:06:30 -0700 PDT
Subject: Re: Exception handling overhead?
Newsgroups: comp.lang.ada

GNAT provides two exception handling mechanisms: zero-cost and setjump/longjump. "Zero-cost" really means zero _distributed_ cost, i.e. you incur a cost only when raising an exception, as you describe. The older setjump/longjump mechanism costs some execution time and memory whether you raise exceptions or not. It is possible to select either mechanism when compiling your program.

For some details, see the short mailing list thread starting at [1].

[1] http://gcc.gnu.org/ml/gcc/2006-10/msg00270.html

Of course, this is compiler-specific.

[…]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 24 Apr 2009 14:42:26 +0200
Subject: Re: Exception handling overhead?
Newsgroups: comp.lang.ada

[…]

Plus, I guess that whatever distributed overhead exceptions might have, it should be negligible comparing to the overhead of unbounded strings.

From: Anonymous
Date: Fri, 24 Apr 2009 14:06:13 GMT
Subject: Re: Exception handling overhead?
Newsgroups: comp.lang.ada

You must disable the compiler default Range checking, like:

```
function Element( B : Buffer; Index :
Positive ) return Character is
  begin
    declare
    -- Temporary disable system checks
    pragma Suppress ( Range_Check );
    begin
      if Index >
         Ada.Strings.Unbounded.Length
           ( B.Internal_String ) then
        raise Bad_Index;
      end if;
    end;
    return Ada.Strings.
           Unbounded.Element (
           B.Internal_String, Index );
  end Element;
```

From: Georg Bauhaus <rm.dash-bauhaus@futureapps.de>
Date: Fri, 24 Apr 2009 17:48:56 +0200
Subject: Re: Exception handling overhead?
Newsgroups: comp.lang.ada

> You must disable the compiler default Range checking, like:

Or Index_Check. However, this only gives an implementation permission to omit certain checks. So I wonder whether pragma Suppress (*_Check) will have an effect on the library code, or whether the

library will have to be recompiled, and then, whether recompilation of the library will still produce correct code.

Programming graphic systems in Ada

From: Olivier Scalbert <olivier.scalbert@algosyn.com>
Date: Thu, 23 Apr 2009 11:27:59 +0200
Subject: Programming graphic systems
Newsgroups: comp.lang.ada

[…]

When playing with my GPS, I was thinking of the beautiful graphic systems used in safety critical environment (medical, aircraft, ATC, …).

I assume lots of them must be written in Ada. I was asking myself how are they programmed mainly at the graphic level. Does the application part use a graphic toolkit provided by the graphic board manufacturer or do you have to program all the graphic primitives yourself in Ada (as Bresenham line algorithm, …) and written to the raster memory (which must be very cool!) ?

[…]

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Thu, 23 Apr 2009 04:54:03 -0700 PDT
Subject: Re: Programming graphic systems
Newsgroups: comp.lang.ada

[…]

For aircrafts, you usually have an OpenGL API to the graphics card, though I'm not aware of any 'safety critical' display systems, DO-178B /Level C is fairly typical. I'm sure there must be Level A/B display systems…

In older systems, you might have a cursive display and primitives provided by your (often) bespoke hardware (e.g. move, draw_to, fill, etc). You then create 'scripts' to draw what you need.

From: Paul Zacharzewski <paul@ipquickly.com>
Date: Thu, 23 Apr 2009 13:17:26 -0600
Subject: Re: Programming graphic systems
Newsgroups: comp.lang.ada

[…]

I remember reading about an X11 system rewritten in Ada, a while ago.

Does anyone else knows something about this?

From: Samuel Tardieu <sam@rfc1149.net>
Date: Fri, 24 Apr 2009 01:55:42 +0200
Subject: Re: Programming graphic systems
Newsgroups: comp.lang.ada

[…]

Are you referring to XInada?

http://www.topgraphx.com/version_am/fichier_prod_xinada.htm

From: Anonymous
Date: Thu, 23 Apr 2009 23:01:42 +0200
Subject: Re: Programming graphic systems
Newsgroups: comp.lang.ada

[…] I'd say glass cockpits are pretty new. At least for civil aircrafts. And my guess is that most would use something like SCADE.

http://www.esterel-technologies.com/products/scade-display/

[…]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 23 Apr 2009 14:31:37 -0700 PDT
Subject: Re: Programming graphic systems
Newsgroups: comp.lang.ada

[…]

Until a couple of years ago, I used to work for Barco avionics [1,2] and indeed most of our devices were programmed in Ada. We wrote our own display drivers in Ada; they worked by writing into the graphics processor's registers over PCI and I2C. For bitmaps and video, we would write into a RAM buffer then point the graphics board to the buffer (i.e. write the address of the buffer in a register of the board).

However, to draw the geometrical symbols for the flight instruments, we would of course use the hardware primitives rather than re-implement them in software.

After writing a command and parameters in a set of registers, we would wait for a completion flag to become true in another register.

[1] http://www.barco.com/aerospace

[2] http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/events/07/070612-abga.html

For air (or rail, or seaport) traffic control, Barco also offers 2048x2048-pixel liquid crystal displays with an X server embedded in them.

I didn't work in that division so I can't provide any more details.

Nowadays I work in air traffic flow management [3] and we use an in-house binding to Motif for older screens and GtkAda for newer ones.

ETFMS is a mission-critical, soft-real-time, distributed application.

It is neither life-critical (i.e. nobody dies if it crashes) nor embedded (i.e. no DO-178B certification is needed) and the real-time requirements are in seconds, not micro- or nanoseconds, so we can use mainstream graphical subsystems (i.e. X11 on "normal" workstations and servers).

[3] http://www.cfmu.eurocontrol.int/cfmu/public/standard_page/developments_etfms_index.html

From: Dimonax
Date: Fri, 24 Apr 2009 19:22:52 GMT
Subject: Re: Programming graphic systems
Newsgroups: comp.lang.ada

[…]

Actually if you know how to write callbacks to a C API with Ada, then getting started isn't all that complicated. There are a few people even in this newsgroup that have written small graphics engines on the OpenGL API.

However if you plan to utilize tasking and such for your programs, you'll need access to the technical documentation of your video hardware, since OpenGL isn't multithreaded and DirectX is pretty much a black box when it comes to serious graphics programming. You might find a graphics card vendor whose drivers also include non-standard support for tasks and threads, but it's unlikely.

Of course I'm referring to the common hardware your likely to find at your local computer shop. The requirements for the big multimillion dollar flight simulators are going to be completely different.

I played with this a few years ago using a Matrox G400 video card and the documentation which, at the time, was freely available. It can be done, and done well, but it takes a lot of patience to get it running.

Also if your running on Linux, you'll find the half-in kernelspace half-in userspace architecture infuriating(one reason I decided to write my own driver.)

The best card to learn the ins and outs of this type of programming, with any language(not just Ada) is one where the tech docs are open. Lately ATI has been pretty forthcoming with their tech docs, so I'd start with them.

Aborting a call to Accept_Socket

From: Tony Truand <truand.tony@gmail.com>
Date: Tue, 21 Apr 2009 08:40:13 -0700 PDT
Subject: Aborting a call to Accept_Socket
Newsgroups: comp.lang.ada

I would like to abort a call to Accept_Socket() if no connection request arrives within a specified time (20 seconds). For me, a quite simple solution (perhaps not safe) is to use an asynchronous transfer of control like this:

```
procedure Server is
   …
begin
  GNAT.Sockets.Initialize;
  …
  loop
  …
   select
     delay 20.0;
     exit;
```

```
    then abort
      GNAT.Sockets.Accept_Socket (…);
    end select;
    …
  end loop;
end Server;
```

The expected behaviour was the end of the program after 20 seconds (if no connection request arrives).

I observe: after 20 seconds the program will terminate only if a connection request arrives. Is this behaviour correct?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 21 Apr 2009 18:21:57 +0200
Subject: Re: Aborting a call to Accept_Socket
Newsgroups: comp.lang.ada

Yes, it is. Asynchronous transfer of control is not guaranteed to work with an outstanding calls. Most likely it does not work as in this case. The behavior is correct because Ada does not know how to abort a socket operation in order to implement this statement. Ada RM contains a list of abort deferred things, which includes potentially any call to any external operation. Specifically for sockets there is a solution: you close the socket from another task. That will kill accept with an error code.

From: Maciej Sobczak <maciej@msobczak.com>
Date: Tue, 21 Apr 2009 14:03:01 -0700 PDT
Subject: Re: Aborting a call to Accept_Socket
Newsgroups: comp.lang.ada

> I would like to abort a call to Accept_Socket() if no connection request arrives within a specified time (20 seconds).

Then you want to use the selector with timeout.

In order to wait for the listening socket put it in the "reading" selector set.

If the Check_Selector finishes with the Status equal to Completed, it means that you can call Accept_Socket without blocking, because there is an incoming connection pending. Otherwise the timeout has expired, which means that there was no incoming connection during the given time.

> For me, a quite simple solution (perhaps not safe) is to use an asynchronous transfer of control

Unfortunately there is no integration of ATC and I/O.

Especially when it comes to non-standard I/O.

From: Anonymous
Date: Tue, 21 Apr 2009 23:24:13 GMT
Subject: Re: Aborting a call to Accept_Socket
Newsgroups: comp.lang.ada

You must use the Selectors routines in the Socket package.

Then set the Timeout parameter to 20 second in procedure Check_Selector.

This routine will wait until an event has occurred. Once returned then you can check the return value for the parameter Status to determine which one of the following three events has occurred: Completed, Expired, or Aborted.

- Complete: one of the expected events occurred
- Expired: no event occurred before the expiration of the timeout
- Aborted: an external action cancelled the wait operation before any event occurred.

Ada on Beagleboard

From: Olivier Scalbert <olivier.scalbert@algosyn.com>
Date: Thu, 30 Apr 2009 08:33:33 +0200
Subject: Ada and beagleboard
Newsgroups: comp.lang.ada

Do you think it is possible to experiment a little of Ada on the beagleboard?

http://beagleboard.org/

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 30 Apr 2009 01:00:30 -0700 PDT
Subject: Re: Ada and beagleboard
Newsgroups: comp.lang.ada

Yes, I think it is possible. There are two options, both requiring a significant amount of work:

1) cross-compilation

You'd build a cross-compiler targeting arm-linux. If your host is Debian, I'd be interested in your patches so that future Debian releases support such a cross-compiler out of the box[1].

[1] http://lists.debian.org/debian-gcc/2009/02/msg00053.html and replies

2) native compilation

It should be possible to install Debian's ARM port on the board.

Unfortunately gnat-4.3 is not currently supported on arm, so you'd have to build it using the cross-compiler. I'd be happy to integrate your patches so that future Debian releases include gnat on arm. The 128 MB of RAM might prove insufficient to run gnat comfortably, especially if the board doesn't support paging.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 30 Apr 2009 06:56:03 -0700 PDT
Subject: Re: Ada and beagleboard
Newsgroups: comp.lang.ada

[…]

> What would be a comfortable amount of RAM in your estimation?

If you want to build GNAT, at least 1 GB is necessary(*).

This requirement would apply, for example, to a Debian "build daemon" which recompiles all packages in Debian natively (including, therefore, GNAT and other large packages).

If you only want to compile your own small programs with GNAT, 512 MB should be sufficient; less if you can use paging to disk; but, again, I don't know whether the BeagleBoard supports paging or not.

(*) I used to use an IBM ThinkPad T22 with a Pentium III@900 MHz and 256 MB RAM. Building GNAT 3.15p on it was OK; building GCC 4.1 with only C and Ada enabled took 7 hours and heavy swapping. GCC 4.3 is even larger. I replaced my T22 with a dual-core laptop with 2 GB RAM which I upgraded to 3 GB for the purposes of recompiling GNAT.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 4 May 2009 00:50:08 -0700 PDT
Subject: Re: Ada and beagleboard
Newsgroups: comp.lang.ada

I just came across an alternative to the BeagleBoard that just might have enough resources to build GNAT: the "plug computer", see

http://plugcomputer.org/

It has 512 MB RAM, 512 MB Flash and a 1.2 GHz ARM processor.

Benchmarking GNAT GPL

From: johnscpg@googlemail.com
Date: Mon, 1 Jun 2009 04:38:24 -0700 PDT
Subject: benchmarking GPL
Newsgroups: comp.lang.ada

In celebration of the arrival of the new GNAT GPL (20090519) I decided to do some benchmarking to see how the optimizer's coming along. I was slightly more than rather pleased with the results. Results in gory detail are appended below.

I compared 6 (smallish) programs, written in both Ada and Fortran. Four of them are in C also. All of the routines can be found at:

http://web.am.qub.ac.uk/users/j.parker/bench_depository/

4 compilers are used:

- gcc 4.3.4,
- Intel Fortran ifort 11.0 (latest version),
- GNAT GPL (20090519),
- gfortran (based on gcc version 4.3.2).

Operating system: Debian Linux, Lenny.

Processor: Intel x86-64 (Xeon X5460 @ 3.16GHz).

The Intel Fortran (ifort) is an aggressive optimizing compiler, especially on numerical linear algebra. Its use here is reassuring: if our gcc-family results on numerical calculations were suboptimal by a large factor, ifort would likely let us know. It also has the easiest optimization flags: it's a simple choice between -O3, -fast, -ipo and a few other things that make almost no difference.

Two of the 6 programs I wrote myself: an FFT benchmark called fft1tst.adb, and a jacobi eigendecomposition called jacobi_eigen_bench_1.adb. Both are accompanied by near identical fortran versions. This exhausted my entire supply of inter-language benchmarking routines, so 4 of the test programs I downloaded from a depository of small benchmarking routines:

http://shootout.alioth.debian.org/gp4/

I downloaded C, Fortran, Ada versions of: nsievebits, nbody, binarytree, mandelbrot.

I made small modifications to 2 of the Ada programs. In one case I degraded the Ada code to a slower, older version so that it was identical to the C version I was comparing with. In the other case I replaced a packed boolean array with an array of unsigned ints. These were the only changes to any of the programs, so the exercise mostly amounted to finding good optimization flags. I tried to find good optimization flags for the C and Fortran compilations too, and managed to speed up a few of them after several attempts. The original Ada test programs were written by Pat Rogers and Pascal Obry. (Thanks!)

Inter-language benchmarks shouldn't be taken too seriously, but I learned a few useful things:

the -mfpmath=387 and -mfpmath=387,sse flags came as real surprize to me. In several cases they made all the difference. I also noticed that GNAT seems to be be doing a better job of optimizing operations on packed boolean arrays, and a better job on some linear algebra problems. Unless I'm mistaken, in the lin alg case (jacobi_eigen below) the improvement over the old days is almost a factor of 2. Thanks GNAT!

[…]

FFT1TST2:

Compilation Commands:

gnatmake fft1tst2.adb -O3 -gnatNp -march=native

gfortran fft1tst2.f -O3 -march=native -o fft1tst2

ifort fft1tst2.f -O3 -WB -xT -o fft1tst2

Execute:

time ./fft1tst2

Running Time (using 8192 data points):

GNAT: 2.748 seconds

gfortran: 2.812 seconds

ifort: 2.816 seconds

Running Time (using 4096 data points):

GNAT: 1.000 seconds

gfortran: 1.004 seconds

ifort: 1.088 seconds

Running Time (using 1024 data points):

GNAT: 0.168 seconds

gfortran: 0.184 seconds

ifort: 0.176 seconds

Notes:

The Ada and the Fortran77 FFT's were written over 17 years ago. Both are Radix 4 fast fourier transforms. These versions were written for benchmarking rather than ultimate speed: the idea was to make the 2 the same wherever possible, for language/compiler comparisons. If I use -ffast-math in the GNAT compilation it runs exactly the same speed as the gfortran (very slightly slower), so I suspect -ffast-math is always used by gfortran.

JACOBI_EIGEN

Compilation Commands:

gnatmake jacobi_eigen_bench_1.adb -O3 -gnatNp -march=native -ffast-math -funroll-loops -o eig

gfortran jacobi_eigen.f90 -O3 -march=native -ffast-math -funroll-loops -o eig

ifort jacobi_eigen.f90 -O3 -ipo -static -o eig

Execute:

time ./eig

Running Time (100x100 matrices (100 iterations)):

GNAT: 1.636 seconds

gfortran: 1.720 seconds

ifort: 1.440 seconds

Running Time (1000x1000 matrices (1 iteration)):

GNAT: 23.7 seconds

gfortran: 39.7 seconds

ifort: 37.9 seconds

Notes:

Matrix size and no of iterations has to be typed in at the top of the 2 routines:

jacobi_eigen.f90, jacobi_eigen_bench_1.adb.

A year ago the GNAT executables for Jacobi were much slower than gfortran, and ifort.

Don't know what happened in the 1000x1000 case, but I am not displeased.

Notice we are using the same compiler flags in the gfortran and GNAT cases.

The number of arithmetical operations performed by these routines is exactly proportional to No_of_Rotations, which is output on completion. The difference between the Fortran No_of_Rotations and the Ada No_of_Rotations is under 3% here.

NBODY:

Compilation Commands:

gnatmake nbody.adb -O3 -gnatNp -march=native -ffast-math -funroll-loops -ftracer -freorder-blocks-and-partition -mfpmath=387,sse

gfortran nbody.f90 -O3 -march=native -funroll-all-loops -o nbody

ifort nbody.f90 -O3 -no-prec-div -o nbody

gcc nbody.c -O3 -o nbody

Execute:

time ./nbody 24000000

Running Time:

GNAT: 4.908 seconds

gfortran: 5.602 seconds

ifort: 4.660 seconds

GCC: 4.472 seconds

Notes:

The obscure compilation flags (-ftracer -freorder-blocks-and-partition) are not needed if you write the inner loop of nbody_pck.Advance a bit differently. I just wanted to use the original version of nbody.adb. nbody2.adb is more like the C version and has simpler optimization flags, but runs at same speed as nbody.adb. The gcc C is about 9% faster than nbody.adb -a small but interesting difference I don't understand.

NSIEVEBITS:

Compilation Commands:

gnatmake nsievebits2.adb -O3 -gnatnp -march=native -funroll-loops -ftracer

gfortran nsievebits.f90 -O3 -march=native -funroll-loops -o nsievebits2

ifort nsievebits.f90 -O3 -ipo -static -o nsievebits2

gcc nsievebits.c -O3 -march=native -funroll-loops -o nsievebits2

Execute:

time ./nsievebits2 11

Running Time:

GNAT: 0.320 seconds

gfortran: 0.388 seconds

ifort: 0.372 seconds

GCC: 0.364 seconds

Notes:

nsievebits2.adb uses an array of unsigned ints to replace the packed boolean array in nsievebits.adb. Both methods could not be

more legitimate in this exercise. GNAT has improved remarkably: the packed boolean array version (nsievebits.adb) is now competitive with the other languages.

MANDELBROT:

Compilation Commands:

gnatmake mandelbrot.adb -O3 -gnatnp -march=native -ffast-math -funroll-loops -mfpmath=387

gfortran mandelbrot.f90 -O3 -march=native -funroll-loops -o mandelbrot

ifort mandelbrot.f90 -O3 -ipo -static -o mandelbrot

gcc mandelbrot.c -O3 -march=native -ffast-math -funroll-loops -mfpmath=387 -o mandelbrot

Execute:

time ./mandelbrot 3000

Running Time:

print-to-screen disabled: (these are meaningful timings.)

GNAT: 0.980 seconds

gfortran: 1.112 seconds

ifort: 1.180 seconds

GCC: 0.960 seconds

[…]

Notes:

[…] The Fortran uses the original complex number implementation of the mandelbrot inner loop. I modified the Ada version was to use exactly the same inner loop as the C version (even though the modification slowed down the Ada version). In both the Ada and the C versions it was the -mfpmath=387 flag (which I assume disables sse) that did the trick of speeding them up.

BINARYTREES:

Compilation Commands:

gnatmake binarytrees.adb -O3 -gnatnp -march=native -ftracer

gfortran binarytrees.f90 -O3 -march=native -o binarytrees

ifort binarytrees.f90 -fast -static -o binarytrees

gcc binarytrees.c -O3 -march=native -lm -o binarytrees

Execute:

time ./binarytrees 16

Running Time (fastest observed):

GNAT: 1.232 seconds

gfortran: 1.084 seconds

ifort: 1.676 seconds

GCC: 1.060 seconds

Notes:

Insensitive to optimization flags.

Unchecked_Union with empty variant

From: Ivan Levashew <octagram@bluebottle.com>
Date: Sun, 15 Mar 2009 19:34:36 +0600
Subject: Unchecked_Union with empty variant
Newsgroups: comp.lang.ada

```
procedure Check_Unions is
  type Complex_Record
    (Kind : Integer := 0) is record
     Constant_Part : Character;
     Constant_Part2 : Character;
     case Kind is
       when 0 =>
            Variant_Part1 : Character;
      when 1 =>
            Variant_Part2 : Character;
      when 2 =>
            Variant_Part3 : Character;
      when others =>
            null;
    end case;
  end record;
  pragma Unchecked_Union
    (Complex_Record);
begin
   null;
end Check_Unions;
```

It gives an error:

C:\…\GEMA-Win32API\Ada_test>gnatmake -gnat05 Check_Unions.adb

gcc -c -gnat05 check_unions.adb

check_unions.adb:14:04: Unchecked_Union may not have empty component list

gnatmake: "check_unions.adb" compilation error

What's the problem?

I can't see any words "empty" or "null" here:

http://www.adaic.com/standards/05aarm/html/AA-B-3-3.html

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Mon, 16 Mar 2009 17:52:54 -0400
Subject: Re: Unchecked_Union with empty variant
Newsgroups: comp.lang.ada

There is 14/2:

All objects of an unchecked union type have the same size.

GNAT may be interpreting that to mean all variants must be the same size. But that's not true, it accepts an Integer:

```
  type Complex_Record
    (Kind : Integer := 0) is record
     Constant_Part : Character;
     Constant_Part2 : Character;
     case Kind is
       when 0 =>
          Variant_Part1 : Character;
       when 1 =>
          Variant_Part2 : Integer;
       when 2 =>
          Variant_Part3 : Character;
       when others =>
          Variant_Part4 : Character;
     end case;
  end record;
```

So it looks like a compiler bug.

From: Ivan Levashew <octagram@bluebottle.com>
Date: Tue, 17 Mar 2009 13:21:02 +0600
Subject: Re: Unchecked_Union with empty variant
Newsgroups: comp.lang.ada

GNAT supported Unchecked_Union in Ada 95 mode. UU wasn't in Ada 95 standard, and GNAT could only handle pure union (without declarative part). Ada 2005 allowed declarative parts.

I have a theory that GNAT developers allowed declarative parts but forgot to disable restriction applicable to records meant to be union mirrors.

Anyway I have to deal with current tools so I defined several Union_n subtypes.

From: Adam Beneschan <adam@irvine.com>
Date: Mon, 16 Mar 2009 08:16:18 -0700 PDT
Subject: Re: Unchecked_Union with empty variant
Newsgroups: comp.lang.ada

[…]

I can't find any rule that makes this illegal. The error message indicates, though, that someone at GNAT thought it was illegal---i.e. it's not simply a "bug" due to incorrectly implementation of the requirements, but was rather a misinterpretation of the standard. But either way, I think the compiler is wrong.

Problem with discriminants and interfaces

From: Reto Buerki <reet@codelabs.ch>
Date: Mon, 09 Mar 2009 18:21:34 +0100
Subject: Discriminant & interface not implemented by full type
Newsgroups: comp.lang.ada

I have the following simple test application:

```
package Full_View is
  type Base_Type (Num : Integer) is
    tagged private;
  type Base_Interface is interface;
  type New_Type is new Base_Type
    (Num => 11) and
    Base_Interface with private;
```

```
private
   type Base_Type (Num : Integer) is
      tagged record
         My_Number : Integer := Num;
      end record;
   type New_Type is new Base_Type
      (Num => 11) and
      Base_Interface with null record;
end Full_View;
```

Trying to compile this with gnat gcc 4.3 in Debian/Lenny results in the following error:

full_view.ads:7:09: interface "Base_Interface" not implemented by full type (RM-2005 7.3 (7.3/2))

When removing the (Num => 11) discriminant initialization or the Base_Interface interface from the New_Type type extension, the code compiles fine.

Is this a compiler bug? Seems to me that the consistency check of the partial and full view of a tagged type covering interfaces does not like discriminants.

[…]

From: Adam Beneschan <adam@irvine.com>
Date: Wed, 11 Mar 2009 18:44:40 -0700 PDT
Subject: Re: Discriminant & interface not implemented by full type
Newsgroups: comp.lang.ada

Sure looks like a compiler bug to me. The program definitely doesn't violate the RM section paragraph that the error message refers to.

From: Per Sandberg <per.sandberg@bredband.net>
Date: Thu, 12 Mar 2009 07:21:08 +0100
Subject: Re: Discriminant & interface not implemented by full type
Newsgroups: comp.lang.ada

Compiles fine with GNATPro 6.2.1

From: Reto Buerki <reet@codelabs.ch>
Date: Thu, 12 Mar 2009 14:07:06 +0100
Subject: Re: Discriminant & interface not implemented by full type
Newsgroups: comp.lang.ada

I filed a GCC bug report [1].

[1] http://gcc.gnu.org/bugzilla/show_bug.cgi?id=39441

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.

2009

☺ July 01-03 21st **Euromicro Conference on Real-Time Systems** (ECRTS'2009), Dublin, Ireland. Topics include: applications (consumer electronics; multimedia and entertainment; process control; avionics, aerospace; automotive; telecommunications); software technologies (compiler support, component-based approaches, middleware and distribution technologies, programming languages and operating systems); system design and analysis (modelling and formal methods, reliability and security in RT systems, scheduling and schedulability analysis, worst-case execution time analysis, validation techniques, ...); etc.

June 30 9th **International Workshop on Worst-Case Execution Time Analysis** (WCET'2009). Topics include: any issue related to timing analysis, in particular Integration of WCET and schedulability analysis; Evaluation, case studies, benchmarks; Tools for WCET analysis; Program design for timing predictability; Integration of WCET analysis in development processes; WCET analysis for multi-threaded and multi-core systems; etc.

☺ June 30 **Workshop on the Definition, evaluation, and exploitation of modelling and computing standards for Real-Time Embedded Systems** (STANDRTS'2009). Topics include: methodologies or tools for the design validation or verification of RTES based on modelling standards; modelling languages and formalisms like UML, SysML, MARTE, EAST-ADL, AUTOSAR, AADL; automatic code generation for current and prospective standard languages like Ada, C, Esterel; etc.

July 01-03 9th **International Conference on Application of Concurrency to System Design** (ACSD'2009), Augsburg, Germany. Topics include: (Industrial) case studies of general interest, gaming applications, consumer electronics and multimedia, automotive systems, (bio-)medical applications, internet and grid computing, ...; Synthesis and control of concurrent systems, (compositional) modelling and design, (modular) synthesis and analysis, distributed simulation and implementation, ...; etc.

July 03-08 14th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2009), Paris, France.

July 05-12 36th **International Colloquium on Automata, Languages and Programming** (ICALP'2009), Rhodes, Greece. Topics include: Parallel and Distributed Computing; 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; etc.

☺ July 06-10 23rd **European Conference on Object Oriented Programming** (ECOOP'2009), Genova, Italy. Topics include: research results or experience in all areas relevant to object technology, including work that takes inspiration from, or builds connections to, areas not commonly considered object-oriented; examples are: 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 06 19th **Doctoral Symposium and PhD Students Workshop**. Topics include: Design Patterns, Concurrency, Real-time, Embeddedness, Distribution, Language Workbenches, Generative Programming, Language Design, Language Constructs, Static Analysis, Language Implementation, Methodology, Practices, Design Languages, Software Evolution, Formal methods, Tools, Programming environments, etc.

☺ July 07 1st **International Workshop on Distributed Objects for the 21st Century** (DO21'2009). Topics include: State-of-the-art distributed object systems; Language abstractions for developing Software as a Service; Combining objects with other paradigms (e.g. events, publish/ subscribe, tuples, dataflow, REST, ...); Alternative (non-OO) approaches to the above (and their pros/cons); etc.

☺ July 07 8th **Workshop on Parallel/High-Performance Object-Oriented Scientific Computing** (POOSC'2009). Topics include: identifying specific problems impeding greater acceptance and widespread use of object-oriented programming in scientific computing; proposed and implemented solutions to these problems; and new or novel approaches, techniques or idioms for scientific and/or parallel computing. Specific areas of interest include: alternatives or extensions to mainstream object-oriented languages (e.g. C++, Java); performance issues and their realized or proposed resolution; issues specific to handling or abstracting parallelism, including the handling or abstraction of heterogeneous and multicore microarchitectures; higher level languages (e.g. domain specific languages) or their embedding into OO languages to support parallelism or specific tasks in scientific computing; grand visions (of relevance); etc.

July 13-16 2009 **International Conference on Software Engineering Theory and Practice** (SETP'2009), Orlando, Florida, 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. Event includes: special session on Object-Oriented Programming.

☺ July 13-16 **International Conference on Parallel and Distributed Processing Techniques and Applications** (PDPTA'2009), Las Vegas, Nevada, USA. Topics include: Parallel/Distributed applications; Reliability and fault-tolerance; Real-time and embedded systems; Software tools and environments for parallel and distributed platforms: operating systems, compilers, languages, debuggers, monitoring tools, software engineering on parallel/distributed systems, ...; Object oriented technology and related issues; Scheduling and resource management; etc.

July 19-23 ACM **International Symposium on Software Testing and Analysis** (ISSTA'2009), Chicago, Illinois, USA.

☺ July 19 2nd **International Workshop on Defects in Large Software Systems** (DEFECTS'2009). 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.

July 19-20 **Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging** (PADTAD'2009). Topics include: Curriculum and education for multi-core design, programming, testing, and analysis; Tools for testing or debugging of Multi-threaded/Parallel/Distributed (MPD) applications; Debugging and testing MPD applications; Using static analysis or formal verification to enhance debugging and testing of MPD applications; Detecting race conditions and deadlocks; Replay of MPD applications; Finding timing bugs early in the process; Testing real-time MPD applications; Testing the fault tolerance of MPD applications; Pilots in applying new testing techniques to MPD applications; Teaching of MPD system design, verification and testing; etc.

July 29-31 3rd IEEE **International Symposium on Theoretical Aspects of Software Engineering** (TASE'2009), Tianjin, China. Topics include: Specification and Verification; Program Analysis; Model-Driven Engineering; Software Architectures and Design; Object Orientation; Embedded and Real-Time Systems; Component-Based Software Engineering; Software Safety, Security and Reliability; Reverse Engineering and Software Maintenance; Type System; Dependable Concurrency; etc.

☺ August 10-12 7th IEEE **International Symposium on Parallel and Distributed Processing with Applications** (ISPA'2009), Chengdu and Jiuzhai Valley, China. Topics include: all aspects of parallel and distributed computing and networking, such as Parallel/distributed system architectures, Tools and environments for software development, Distributed systems and applications, Reliability, fault-tolerance, and security, etc.

August 10-13 28th **Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing** (PODC'2009), Calgary, Alberta, Canada.

☺ August 19-21 13th **Brazilian Symposium on Programming Languages** (SBLP'2009), Gramado, Rio Grande do Sul, Brazil. Topics include: Programming language design and implementation, Design and implementation of programming language environments, Object-oriented programming languages, Program transformations, Program analysis and verification, Compilation techniques, etc.

August 24-28 7th **Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering** (ESEC/FSE'2009), Amsterdam, the Netherlands. Topics include: Specification and verification, Software architecture and design, Tools and environments, Software quality and performance, Formal methods, Component-based software engineering, Distributed systems and middleware, Embedded and real-time systems, Open standards and certification, Dependability (safety, security, reliability), Case studies and experience reports, etc. Deadline for early registration: July 19, 2009.

☺ August 25-28 15th **International European Conference on Parallel and Distributed Computing** (Euro-Par'2009), Delft, the Netherlands. Topics include: all aspects of parallel and distributed computing, such Support tools and environments, High performance architectures and compilers, Distributed systems and algorithms, Parallel and distributed programming, Multicore and manycore programming, Theory and algorithms for parallel computation, etc.

- August 24 2nd **Workshop on Productivity and Performance** (PROPER'2009). Topics include: tools and tool approaches for parallel program development and analysis; success stories about optimization or parallel scalability achieved using tools; etc.
- ☺ August 25 3rd **Workshop on Highly Parallel Processing on a Chip** (HPPC'2009). Topics include: programming models, languages and software libraries, implementation techniques, support and performance tools, performance evaluation, parallel algorithms and applications, migration of existing codebase, teaching of parallel computing, for/on highly parallel multi-core systems.

☺ Aug 31 – Sep 04 10th **International Conference on Parallel Computing Technologies** (PaCT'2009), Novosibirsk, Russia. Topics include: New developments, applications, and trends in parallel computing technologies; All aspects of the applications of parallel computer systems; Languages, environment and software tools supporting parallel processing; General architecture concepts; Teaching parallel processing; etc.

☺ September 01-04 **International Conference on Parallel Computing 2009** (ParCo'2009), Lyon, France. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments. Deadline for early registration: July 15, 2009.

☺ September 01-04 4th **Latin-American Symposium on Dependable Computing** (LADC'2009), João Pessoa, Brazil. Topics include: Dependability of software (analysis, architecture, testing, verification & validation, software certification); Dependability of maintenance; Security; Dependability and human issues; Safety; etc.

September 01-05 20th **International Conference on Concurrency Theory** (CONCUR'2009), Bologna, Italy. Topics include: concurrency theory and its applications, e.g. semantics, cross-fertilization between industry and academia, etc. Deadline for early registration: July 31, 2009.

September 09-11 8th **International Conference on Software Methodologies, Tools, and Techniques** (SoMeT'2009), Prague, Czech Republic. 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; Software Engineering models, and formal techniques for software representation, software testing and validation; etc.

☺ Sep 12-16 18th **International Conference on Parallel Architectures and Compilation Techniques** (PACT'2009), Raleigh, North Carolina, USA. Topics include: Parallel computational models; Compilers and tools for parallel computer systems; Support for concurrency correctness in hardware and software; Parallel programming languages, algorithms and applications; Middleware and run-time system support for parallel computing; Reliability and fault tolerance for parallel systems; Modeling and simulation of parallel systems and applications; Parallel applications and experimental systems studies; etc.

September 14-17 Joint 8th **Working International Conference on Software Architecture** and 3rd **European Conference on Software Architecture** (WICSA/ECSA'2009), Cambridge, UK. Topics include: architecture description languages; architecture reengineering, discovery and recovery; software architects' roles and responsibilities, training, education and certification; etc.

September 15 **Safecomp2009 - Workshop on the Design of Dependable Critical Systems** (DDCS'2009), Hamburg, Germany. Topics include: Automotive and aerospace, Robotics, etc. Deadline for submissions: July 30, 2009 (papers, demos).

September 16-18 12th **International Conference on Quality Engineering in Software Technology** (CONQUEST'2009), Nuremberg, Germany. Topics include: specific real-life case studies with detailed quality analysis and evaluation; quality engineering issues in domains such as Medical IT, Automotive, Avionics, Transport, and IT; etc.

☺ Sep 22-25 38th **International Conference on Parallel Processing** (ICPP'2009), Vienna, Austria. Topics include: Programming Models, Languages, and Compilers: from high-level abstractions to efficient code, etc.

☺ Sep 27-30 28th IEEE **International Symposium on Reliable Distributed Systems** (SRDS'2009), Niagara Falls, New York, USA. Topics include: Security and high-confidence systems, Safety-critical systems and critical infrastructures, Fault-tolerance in embedded systems, Analytical or experimental evaluations of dependable distributed systems, Formal methods and foundations for dependable distributed computing, etc. Deadline for submissions: July 6, 2009 (workshop papers).

☺ Sep 28 – Oct 02 4th **Working Conference on Programming Languages** (ATPS'2009), Luebeck, Germany. Topics include: All paradigms of programming languages (imperative, object-oriented, functional, logic, concurrent, parallel, or graphical programming languages) as well as languages to support the implementation of distributed systems and concepts for the integration of different paradigms; Design of programming languages as well as domain-specific languages; Implementation and optimization techniques; Analysis and transformation of programs; Type systems; Modelling languages, object orientation; Verification of programs and implementations; Tools and programming environments; Experiences with specific applications; Techniques, methods, concepts, and tools to improve the safety and reliability of programs; etc.

October 04-09 ACM/IEEE 12th **International Conference on Model Driven Engineering Languages and Systems** (MoDELS'2009), Denver, Colorado, USA. Topics include: Development of domain-specific modeling languages, Tools and meta-tools for modeling languages and model-based development, Evolution of modeling languages and models, Experience stories in general (successful and unsuccessful), Issues related to current model-based engineering standards, Experience with model-based engineering tools, etc.

☺ October 06 2nd **International Workshop on Model Based Architecting and Construction of Embedded Systems** (ACES-MB'2009). 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), super-synchronous models (TTA, Giotto), scheduling-friendly models (HRT-UML, Ada Ravenscar), etc. Deadline for submissions: July 20, 2009.

October 05-06 2nd **International Conference on Software Language Engineering** (SLE'2009), Denver, Colorado, USA. Topics include: the engineering of artificial languages used in software development including general-purpose programming languages, domain-specific languages, modeling and meta-modeling languages, data models, and ontologies. Deadline for submissions: July 3, 2009 (abstracts), July 10, 2009 (papers).

October 07-09 14th **International Real-Time Ada Workshop** (IRTAW'2009), Portovenere, Italy. In cooperation with Ada-Europe.

☺ October 11 5th **Workshop on Programming Languages and Operating Systems** (PLOS'2009), Big Sky, MT, USA. Topics include: critical evaluations of new programming language ideas in support of OS construction; type-safe languages for operating systems; language-based approaches to crosscutting system concerns, such as security and run-time performance; language support for system verification; the use of OS abstractions and techniques in language runtimes; etc.

☺ October 12-14 **IMCSIT2009 - 2nd Workshop on Advances in Programming Languages** (WAPL'2009), Mragowo, Poland. Topics include: Compiling techniques; Domain-specific languages; Formal semantics and syntax; Generative and generic programming; Languages and tools for trustworthy computing; Language concepts, design and implementation; Metamodeling and modeling languages; Model-driven engineering languages and systems; Practical experiences with programming languages; Program analysis, optimization and verification; Program generation and transformation; Programming tools and environments; Proof theory for programs; Specification languages; Type systems; etc.

October 13-16 16th **Working Conference on Reverse Engineering** (WCRE'2009), Lille, France. Topics include: all areas of software maintenance, evolution, reengineering, and migration, such as 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, Reengineering patterns, Program transformation and refactoring, etc.

☺ October 25-29 24th **Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2009), Orlando, Florida, USA. Topics include: the intersection between programming languages and software engineering; key programming models and programming methods and related software engineering ideas, technologies, tools, and applications; critical evaluation of accepted practices, proposals for new programming models, exploration and extension of well-established models, and other novel approaches to building systems; etc. Deadline for submissions: July 2, 2009 (posters, demonstrations, student research competition, doctoral symposium, onward! films, student volunteers).

- ☺ October 25 8th **"Killer Examples" workshop**. Topics include: examples that expose bad practice and so lead to better appreciation of good practice, as obtained by following sound object-oriented principles. Deadline for submissions: September 8, 2009.
- October 25-29 **Onward! 09**. Topics include: different ways of thinking about, approaching, and reporting on programming languages and software engineering research.
- October 26 3rd **Workshop on Assessment of Contemporary Modularization Techniques** (ACoM.09). 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 21, 2009 (abstracts), August 28, 2009 (papers).

Oct 30 – Nov 07 16th **International Symposium on Formal Methods** (FM'2009), Eindhoven, the Netherlands. Theme: "Theory meets practice". Topics include: every aspect of the development and application of formal methods for the improvement of the current practice on system developments; of particular interest are papers on tools and industrial applications; etc.

- November 04 8th **International Workshop on Parallel and Distributed Methods in Verification** (PDMC'2009). Topics include: multi-core/distributed model checking, slicing and distributing the state space, parallel/distributed theorem proving and constraints solving, tools and case studies, industrial applications, etc. Deadline for submissions: August 1, 2009 (abstracts), August 7, 2009 (papers).
- November 06 2nd **International Conference on Teaching Formal Methods** (TFM'2009). Topics include: experiences of teaching FMs, both successful and unsuccessful; educational resources including the use of books, case studies and the internet; the advantages of FM-trained graduates in the workplace; changing attitudes towards FMs in students, academic staff and practitioners; etc.

♦ Nov 01-05 ACM **Annual International Conference on Ada and Related Technologies** (SIGAda'2009), St. Petersburg, Tampa Bay area, Florida, USA. Sponsored by ACM

SIGAda, in cooperation with SIGCAS, SIGCSE, SIGPLAN, Ada-Europe, and Ada Resource Association. Deadline for submissions: October 16, 2009 (grants for educators).

☺ November 02-03 14th **International Workshop on Formal Methods for Industrial Critical Systems** (FMICS'2009), Eindhoven, the Netherlands. Topics include: Design, specification, code generation and testing based on formal methods; 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; Impact of the adoption of formal methods on the development process and associated costs; Application of formal methods in standardization and industrial forums; etc.

November 16-19 20th **International Symposium on Software Reliability Engineering** (ISSRE'2009), Mysuru-Bengaluru, India. Topics include: Reliability, availability, and safety of software systems; Validation, verification, and testing; Software quality and productivity; 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.

☺ Nov 23-27 7th IEEE **International Conference on Software Engineering and Formal Methods** (SEFM'2009), Hanoi, Vietnam. Topics include: formal methods technology transfer; software specification, verification and validation; component-based development; programming languages and type theory; program analysis; real-time, hybrid and embedded systems; safety-critical and fault-tolerant systems; software architectures and their description languages; light-weight formal methods; CASE tools and tool integration; applications of formal methods and industrial case studies; etc.

☺ Dec 08-11 15th IEEE **International Conference on Parallel and Distributed Systems** (ICPADS'2009), Shenzhen, China. Topics include: Parallel and Distributed Applications and Algorithms, Multi-core and Multithreaded Architectures, Resource Management and Scheduling, Security, Dependable and Trustworthy Computing and Systems, Real-Time Systems, etc.

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

2010

☺ January 20-22 37th ACM SIGPLAN-SIGACT **Symposium on Principles of Programming Languages** (POPL'2010), Madrid, Spain. Topics include: all aspects of programming languages and systems, with emphasis on how principles underpin practice. Deadline for submissions: July 8, 2009 (abstracts), July 15, 2009 (papers).

☺ February 03-04 2nd **International Symposium on Engineering Secure Software and Systems** (ESSoS'2009), Pisa, Italy. Topics include: security architecture and design for software and systems, systematic support for security best practices, programming paradigms for security, processes for the development of secure software and systems, etc. Deadline for submissions: September 15, 2009 (abstracts), September 30, 2009 (papers), October 24, 2009 (tutorials).

February 17-19 18th **Euromicro International Conference on Parallel, Distributed and network-based Processing** (PDP'2010), Pisa, Italy. Topics include: Parallel Computer Systems (embedded parallel and distributed systems, fault-tolerance, multi/many core systems, ...); Models and Tools for Parallel Programming Environments: Advanced Applications (numerical applications with multi-level parallelism, real time distributed applications, distributed business applications, ...); Languages, Compilers and Runtime Support Systems (parallel languages, object-oriented languages, dependability issues, scheduling, ...); etc. Deadline for paper submissions: July 20, 2009.

☺ March 10-13 41st ACM **Technical Symposium on Computer Science Education** (SIGCSE'2010), Milwaukee, Wisconsin, USA.

March 15-18 14th **European Conference on Software Maintenance and Reengineering** (CSMR'2010), Madrid, Spain. Topics include: Experience reports and empirical studies on maintenance, reengineering, and evolution; Description of education-related issues to evolution, maintenance and reengineering; Mechanisms and techniques for reengineering systems as services; etc. Deadline for submissions: October 9, 2009 (abstracts), October 16, 2009 (full papers), October 23, 2009 (workshops, industry track submissions, tool demonstrations, doctoral symposium, European Projects track).

March 20-25 **European Joint Conferences on Theory and Practice of Software** (ETAPS'2009), Paphos, Cyprus. Events include: FOSSACS, Foundations of Software Science and Computation Structures; FASE, Fundamental Approaches to Software Engineering; ESOP, European Symposium on Programming; CC, International Conference on Compiler Construction; TACAS, Tools and Algorithms for the Construction and Analysis of Systems.

March 22-26 25th ACM **Symposium on Applied Computing** (SAC'2010), Sierre and Lausanne, Switzerland.

☺ Mar 22-26 **Track on Object-Oriented Programming Languages and Systems** (OOPS'2010). Topics include: Language design and implementation; Type systems, static analysis, formal methods; Integration with other paradigms; Components and modularity; Distributed, concurrent or parallel systems; Interoperability, versioning and software adaptation; etc. Deadline for submissions: September 8, 2009 (full papers).

☺ Mar 22-26 **Track on Software Engineering** (SE'2010). Topics include: technologies, theories, and tools used for producing highly dependable software more effectively and efficiently; such as Safety and Security; Dependability and Reliability; Fault Tolerance and Availability; Architecture, Framework, and Design Patterns; Standards; Maintenance and Reverse Engineering; Verification, Validation, and Analysis; Formal Methods and Theories; Component-Based Development and Reuse; Empirical Studies, and Industrial Best Practices; etc. Deadline for submissions: September 8, 2009 (full papers).

☺ Mar 22-26 **Track on Real-Time Systems** (RTS'2010). Topics include: all aspects of real-time systems design, analysis, implementation, evaluation, and case-studies, including scheduling and schedulability analysis; worst-case execution time analysis; modeling and formal methods; validation techniques; reliability; compiler support; component-based approaches; middleware and distribution technologies; programming languages and operating systems; embedded systems; etc. Deadline for submissions: September 8, 2009 (full papers).

Mar 22-26 **Track on Software Verification and Testing** (SVT'2010). Topics include: development of technologies to improve the usability of formal methods in software engineering, tools and techniques for verification of large scale software systems, real world applications and case studies applying software verification, static and run-time analysis, correct by construction development, software certification and proof carrying code, etc. Deadline for submissions: September 8, 2009 (papers).

April 06-09 3rd IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2010), Paris, France. Topics include: Verification & validation, Quality assurance, Empirical studies, Inspections, Tools, Embedded software, Novel approaches to software reliability assessment, etc. Deadline for submissions: September 25, 2009 (abstracts, workshops), October 2, 2009 (full papers).

April 13-16 5th **European Conference on Computer Systems** (EuroSys'2010), Paris, France. Topics include: various issues of systems software research and development, such as systems aspects of Dependable computing, Distributed computing, Parallel and concurrent computing, Programming-language support, Real-time and embedded computing, Security, etc. Deadline for submissions: October 19, 2009.

☺ May 02-08 32nd **International Conference on Software Engineering** (ICSE'2010), Cape Town, South Africa. Topics include: Engineering of distributed/parallel software systems; Engineering of embedded and real-time software; Engineering secure software; Patterns and frameworks; Programming languages; Reverse engineering and maintenance; Software architecture and design; Software components and reuse; Software dependability, safety and reliability; Software economics and metrics; Software tools and development environments; Theory and formal methods; etc. Deadline for submissions: September 6, 2009 (technical/research papers); September 17, 2009 (workshops); October 5, 2009 (tutorials,

education papers, software engineering in practice track); November 26, 2009 (doctoral symposium Papers); January 7, 2010 (research demonstration papers, new and emerging results papers).

♦ June 14-18 15th **International Conference on Reliable Software Technologies - Ada-Europe'2010**, Valencia, Spain. Sponsored by Ada-Europe, in cooperation with ACM SIGAda (approval pending). Deadline for submissions: November 16, 2009 (papers, tutorials, workshops), January 11, 2010 (industrial presentations).

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

Association for Computing Machinery

SIGAda 2009 Advance Program

ACM Annual International Conference on Ada and Related Technologies: Engineering Safe, Secure, and Reliable Software

Hilton St. Petersburg Bayfront Hotel
St. Petersburg (Tampa Bay area), Florida, USA
November 1-5, 2009

Sponsored by the ACM Special Interest Group on the Ada Programming Language (SIGAda)
in cooperation with Ada-Europe, Ada Resource Association, and ACM Special Interest Groups on Embedded Systems, Programming Languages, Computers and Society, and Computer Science Education

Special Keynote Presentations

Echo: A New Approach to Formal Verification Based on Ada The Technology, and Experience in Security and Medical Devices
John Knight
University of Virginia
Computer Science Department

Ada Through the Eyes of Developing Large, Mature, Reliable Systems
Richard Schmidt
Lockheed Martin
Information Systems & Global Services - Civil Group

A Look at Ada from Both Sides Now:
Government and Defense Contractor Perspectives
J.C. Smart
Raytheon
Intelligence and Information Systems

Corporate Sponsors – Platinum

AdaCore
The GNAT Pro Company

Corporate Sponsors – Silver

Ellidiss Software
TNI Europe Limited

LOCKHEED MARTIN

VISIT http://www.sigada.org/conf/sigada2009 TODAY
FOR UP-TO-THE-MINUTE TUTORIAL, PROGRAM, AND EXHIBIT INFORMATION
AND ONLINE REGISTRATION

SIGAda 2009 VENUE & HOTEL

Hilton St. Petersburg Bayfront
333 First Street South
St. Petersburg, Florida 33701 (USA)
Tel: 1-727-894-5000
Fax: 1-727-823-4797

The SIGAda 2009 Conference will be held at the Hilton St. Petersburg Bayfront Hotel. This is a beautiful spot located directly on Tampa Bay, Florida. The weather in November is typically in the high 70's F (20 degrees C) with minimal rain fall. It is easily accessible from either the Tampa or Clearwater/St Petersburg airports. And it is about 1-1/2 hours away from the Orlando International Airport and its associated theme parks. The Tampa Bay area also has the Busch Gardens theme park, the Salvador Dali museum, and numerous beaches on the Gulf coast.

The hotel has reserved a block of rooms for the SIGAda 2009 conference. The conference rate is $95 for single or double occupancy rooms, $105 for triple occupancy rooms, and $115 for quadruple occupancy rooms. A 12% tax will be added per night. All reservations must be guaranteed by credit card. Reservations must be received by October 1, 2009. For further information, see http://www.sigada.org/conf/sigada2009/hotel-rates.html

EXHIBITORS

SIGAda 2009 will include vendor participation, featuring presentations on their products and services. For specific information, please contact the Exhibits Chair, **Alok Srivastava**, Northrup Grumman, *Alok.Srivastava@AUATAC.com*.

GRANTS TO EDUCATORS

As in past years, SIGAda is offering grants to educators to attend the conference. Grants cover the registration and tutorial fees; travel funds are not covered, but members of the GNAT Academic Program may be eligible for travel funds from AdaCore. Applications must be sent by e-mail, no later than October 16, 2009. Grantees will receive instructions for on-line registration, which will be accepted through October 28, 2009. Grant program details are available from the conference website or **Prof. Michael B. Feldman**, *mfeldman@gwu.edu*.

WORKSHOPS / BOFS

Focused workshops are important in shaping Ada technology to better meet the needs of the Ada community. Workshops are free for those registered for the conference. Workshop descriptions are listed at the SIGAda 2009 website. Additional workshops or Birds-of-a-Feather (BoF) sessions are welcome. Workshops have a focused objective and result in a report to be published in *ACM Ada Letters*. BoFs are informal discussion groups. If you would like to propose a Workshop or BoF, please contact the Workshops Chair, **Bill Thomas**, *BThomas@MITRE.Org*

CONFERENCE TEAM

Conference Chair
Greg Gicca
AdaCore
gicca@adacore.com

Program Chair
Jeff Boleng, Lt Col, USAF
US Air Force Academy
jeff.boleng@usafa.edu

Workshops Chair
Bill Thomas
The MITRE Corporation
BThomas@MITRE.org

Exhibits Chair
Alok Srivastava
Northrop Grumman
Alok.Srivastava@AUATAC.Com

Local Arrangements Chair
Currie Colket
colket@acm.org

Tutorial Chair
Richard Riehle
Naval Postgraduate School
rdriehle@nps.edu

Webmaster & Proceedings Chair
Clyde Roby
Institute for Defense Analyses
ClydeRoby@ACM.Org

Publicity Co-Chair
Michael Feldman
George Washington University (ret.)
mfeldman@gwu.edu

Publicity Co-Chair
Ron Price
Softcrafts
softcrafts@aol.com

SIGAda Chair
Ricky E. Sward
The MITRE Corporation
rsward@mitre.org

SIGAda and Conference Treasurer
Geoff Smith
Lightfleet Corporation
gsmith@lightfleet.com

SIGAda Vice Chr, Meetings/Conferences
Alok Srivastava
Northrop Grumman
Alok.Srivastava@AUATAC.Com

SIGAda International Representative
Dirk Craeynest
K.U. Leuven (Belgium)
dirk.craeynest@cs.kuleuven.be

Registration Chair
Thomas A. Panfil
US Department of Defense
Phone and FAX +1 301-498-7313
Office Phone: +1 410 854-5818
tapanfil@acm.org

Preliminary Call for Papers
15th International Conference on Reliable Software Technologies – Ada-Europe 2010
14-18 June 2010, Valencia, Spain

http://www.ada-europe.org/conference2010.html

Conference Chair

Jorge Real
Universidad Politécnica de Valencia, Spain
jorge@disca.upv.es

Program Co-Chairs

Jorge Real
Universidad Politécnica de Valencia, Spain
jorge@disca.upv.es

Tullio Vardanega
University of Padua, Italy
tullio.vardanega@math.unipd.it

Tutorial Chair

Albert Llemosí
Universitat de les Illes Balears, Spain
albert.llemosi@uib.cat

Exhibition Chair

Ahlan Marriott
White Elephant GmbH, Switzerland
Ada@white-elephant.ch

Industrial Chair

Erhard Plödereder
University of Stuttgart, Germany
ploedere@informatik.uni-stuttgart.de

Publicity Chair

Dirk Craeynest
Aubay Belgium & K.U.Leuven, Belgium
Dirk.Craeynest@cs. kuleuven.be

In cooperation with
ACM SIGAda
(approval pending)

General Information

The 15th International Conference on Reliable Software Technologies – Ada-Europe 2010 will take place in Valencia, Spain. Following its traditional style, the conference will span a full week, including a three-day technical program and vendor exhibition from Tuesday to Thursday, along with parallel tutorials and workshops on Monday and Friday.

Schedule

16 November 2009	Submission of regular papers, tutorial and workshop proposals
11 January 2010	Submission of industrial presentation proposals
1 February 2010	Notification of acceptance to all authors
1 March 2010	Camera-ready version of regular papers required
10 May 2010	Industrial presentations, tutorial and workshop material required
14-18 June 2010	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 representing industry, academia and government organizations active in the promotion and development of reliable software technologies. To gather experience on the latest periodic revision of the Ada language standard, contributions that present and discuss the potential of the revised language are especially welcome.

All prospective contributions, whether regular papers, industrial presentations, tutorials or workshops, should address the topics of interest to the conference, which for this edition include but are not limited to:

- **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.
- **Software Architectures**: Design Patterns, Frameworks, Architecture-Centered Development, Component and Class Libraries, Component-based Design and Development.
- **Enabling Technologies**: Software Development Environments, Compilers, Debuggers, Run-time Systems, Middleware Components, Concurrent and Distributed Programming, Ada Language and Technology.
- **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, Languages Vulnerabilities.
- **Embedded Systems**: Multicore Architectures, Architecture Modeling, HW/SW Co-Design, Reliability and Performance Analysis.
- **Mainstream and Emerging Applications**: Manufacturing, Robotics, Avionics, Space, Health Care, Transportation, Energy, Games and Serious Games, etc.
- **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.

Program Committee

Alejandro Alonso, Universidad Politécnica de Madrid, Spain
Ted Baker, Florida State University, USA
John Barnes, John Barnes Informatics, UK
Johann Blieberger, Technische Universität Wien, Austria
Jørgen Bundgaard, Rovsing A/S, Denmark
Bernd Burgstaller, Yonsei University, Korea
Alan Burns, University of York, UK
Rod Chapman, Praxis High Integrity Systems, UK
Dirk Craeynest, Aubay Belgium & K.U.Leuven, Belgium
Alfons Crespo, Universidad Politécnica de Valencia, Spain
Juan A. de la Puente, Universidad Politécnica de Madrid, Spain
Raymond Devillers, Université Libre de Bruxelles, Belgium
Franco Gasperoni, AdaCore, France
Michael González Harbour, Universidad de Cantabria, Spain
José Javier Gutiérrez, Universidad de Cantabria, Spain
Andrew Hately, Eurocontrol CRDS, Hungary
Peter Hermann, Universität Stuttgart, Germany
Jérôme Hugues, Telecom Paris, France
Hubert Keller, Institut für Angewandte Informatik, Germany
Albert Llemosí, Universitat de les Illes Balears, Spain
Kristina Lundqvist, Mälardalen University, Sweden & MIT, USA
Franco Mazzanti, ISTI-CNR Pisa, Italy
John McCormick, University of Northern Iowa, USA
Julio Medina, Universidad de Cantabria, Spain
Stephen Michell, Maurya Software, Canada
Javier Miranda, Universidad Las Palmas de Gran Canaria, Spain
Daniel Moldt, University of Hamburg, Germany
Laurent Pautet, Telecom Paris, France
Luís Miguel Pinho, Polytechnic Institute of Porto, Portugal
Erhard Plödereder, Universität Stuttgart, Germany
Jorge Real, Universidad Politécnica de Valencia, Spain
Alexander Romanovsky, University of Newcastle upon Tyne, UK
Jean-Pierre Rosen, Adalog, France
Sergio Sáez, Universidad Politécnica de Valencia, Spain
Ed Schonberg, AdaCore, USA
Theodor Tempelmeier, Univ. of Applied Sciences Rosenheim, Germany
Jean-Loup Terraillon, European Space Agency, The Netherlands
Santiago Urueña, Grupo de Mecánica de Vuelo, Spain
Tullio Vardanega, Università di Padova, Italy
Francois Vernadat, LAAS-CNRS & INSA Toulouse, France
Daniel Wengelin, Saab, Sweden
Andy Wellings, University of York, UK
Jürgen Winkler, Friedrich-Schiller-Universität, Germany
Luigi Zaffalon, University of Applied Sciences, W. Switzerland

Industrial Committee

To be announced

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 Chairs*.

Proceedings

The conference proceedings will be published in the Lecture Notes in Computer Science (LNCS) series by Springer, and will be available at the start of the conference. 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 1 March 2010. For format and style guidelines authors should refer to the following URL: *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 is ranked class A in the CORE ranking and is listed among the top quarter of CiteSeerX Venue Impact Factor.

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 11 January 2010. 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 10 May 2010, aiming at a 20-minute talk. The authors of accepted presentations will be invited to submit corresponding articles 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 authors of accepted 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 at either end 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 Exhibitors

The commercial exhibition 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.

Ada Conference UK 2009

K Fairlamb

AdaCore, 46 rue d'Amsterdam, Paris 75009, France

Abstract

The Ada Conference UK 2009, operated by the Centre for Software Reliability (CS), took place this year on 24 March at the Church House Conference Centre, Westminster, London.

1 Introduction

The 24th of March was a significant date in the Ada calendar this year, not least because it was Ada Lovelace Day, "an international day of blogging to draw attention to women excelling in technology" according to http://findingada.com/.

It was also, of course, the date of the 2009 edition of the Ada Conference UK, a biennial event showcasing the many and varied uses of the Ada programming language in industry and academia worldwide.

Following two previous editions of the event held in Manchester, the organisers this year decided to move the 2009 event south to London, to the prestigious Church House Conference Centre, which sits in the shadows of the Houses of Parliament and the famous towers of Westminster Abbey.

Figure 1 Conference Location

Whether drawn primarily by the venue or by the programme of speakers, the event successfully managed to live up to its reputation as one of the largest professional Ada events in the world, attracting around 100 attendees from both the UK and abroad, proving that Ada is more than ever at the front of software developers' minds as a language which provides answers for many of today's most complex programming challenges – especially in the areas of real time, embedded and safety-critical applications and in particular as the need for robust and reliable software systems increases.

The day was described by Tom Anderson of the CSR as a "Lockheed Martin sandwich", with thought-provoking opening and closing keynotes by Jim Sutton (of Lean Programming fame) and Judith Klein (Lockheed Martin Fellow and Certified Systems Architect). The technical track offered talks on a diverse range of subjects by leading industrial experts, the abstracts of which are provided below and videos of which can be found on the AdaCore website at www.adacore.com.

In addition to the technical track, a stream of well-attended vendor talks ran in parallel to the technical talks and a broad range of leading Ada toolset and service vendors displayed their technologies in the magnificent Church House Assembly Hall.

The next Ada UK conference is already in preparation, so look out for a forthcoming announcement regarding dates and venue!

2 Conference papers at Ada UK

- **Selecting a Programming Language, The Modern Way**

  Jim Sutton, Lockheed Martin (USA)

  The choice of programming language has a significant impact on project performance. Key business indicators like productivity, quality, customer satisfaction, and insulation from legal challenges are all impacted. Even so, projects typically pick their language(s) by personal preference. Better languages are often overlooked because no one knew how the available options addressed their specific needs. Even projects in the domains for which Ada was designed sometimes settle on languages long ago shown to degrade critical success factors, like integrity, integration and verification.

  Projects do not set out to shoot themselves in the feet. There has been no accepted logical approach for selecting a programming language. Being a highly emotional subject, the easiest way to keep the peace has been to bow to the loudest voice. What projects need is an objective process for identifying the best language for their needs. That also provides them with a defense for their choice (including from legal "due diligence" claims). The systems engineering model provides an accountable process for selecting alternatives, called the "trade study". Lean Production focuses processes of all sorts on their stakeholders' most-important concerns. Combining the two yields a

way for projects to select the language most advantageous to their needs and business performance.

While increasing Ada's usage is not the goal of adopting such a process, the objective way in which Ada itself was developed makes it a highly likely outcome.

- **Experiences with SPARK Ada on an 8-bit Embedded Microcontroller**

  Damian Curtis & David Berry, AWE (UK)

  There is a need for high integrity software for small embedded systems. The Control Systems Group at AWE has developed a route to provide demonstrators of high integrity software systems on an 8-bit microcontroller. This presentation concerns the successes that AWE has had in demonstrating the use of SPARK and Ada on the AVR microcontroller.

  The presentation will cover the background to AWE's software development approach, the successful demonstration of the approach (with a comparison to an equivalent development in C) and an outline of other demonstrator systems where this approach is to be applied.

- **Using Static Analysis as part of Code Review**

  Tucker Taft, SofCheck (USA)

  As static analysis tools have become more sophisticated, their role in the software development process has become a subject of debate. Can the use of a static analysis tool substitute for other presumably more labor-intensive steps in the normal process of coding, testing, verification, validation, and ultimately possibly certification?

  This talk will address the issue of using static analysis as an aid to source code review, which is a labor intensive component of many phases of disciplined software development, from initial coding, through certification. Particular attention will be paid to reviewing code according to specific safety-critical software development standards such as DO-178B, or toward identifying specific security vulnerabilities, such as those identified within the Common Weakness Enumeration (CWE). We will identify specific features of a static analysis tool, such as the automatic extraction of pre-and postconditions from the code itself, that can facilitate productive code review, both for the relatively informal reviews that take place during the coding phase, and for the more formal reviews that may take place later in the development cycle.

- **Issues from the Cassini Project**

  Stefan Helfert, MPI-K, Heidelberg, Germany

- **Tokeneer: An Open-Source Demonstration of High-Assurance Software Engineering**

  Janet Barnes, Praxis high Integrity Systems

  The Tokeneer ID Station (TIS) project was commissioned by the NSA as a demonstration vehicle to show exactly how the Praxis Correctness by Construction development approach matches up to the Common Criteria requirements for Evaluation Assurance Level 5 and to measure the productivity and defect rates under controlled conditions. The project developed this security application used a number of rigorous techniques including formalization of the specification in the Z notation, the use of SPARK as the development language and proof of adherence to the security properties. The whole TIS project is now open source, allowing Praxis to show the wider software community how we develop software to high-assurance levels in a cost effective manner.

  In this presentation I will explain what the Tokeneer ID Station is and place it in the wider context of the Tokeneer system. I will describe the process by which Praxis undertook the development and explain the material that is now available open-source in the context of the process.

- **Leveraging the power of UML2 Ports – a Strategy for their Implementation in Ada**

  Fraser Chadburn, IBM Rational Software

  UMLI.x was great for visualizing how classes are coupled. A key enhancement of UML2 was to improve support for modelling large-scale systems, including the ability to hierarchically decompose systems into parts with clear ports and interfaces. However, since a UML port can be contracted to provide any number of interfaces in UML2, Ada 95 presents a number of implementation challenges when mapping UML to code. This presentation will illustrate how these challenges are elegantly resolved by exploiting Ada 2005 interfaces. It will show how the full power of UML2 ports and components and the ease by which the components can be re-used in multiple assemblies by getting Rhapsody to automatically generate the wiring code.

- **Project Coverage and the Open-DO Initiative**

  Franco Gasperoni, AdaCore (France)

  In this presentation we introduce Project Coverage, the next generation of Open Source code coverage tools, offering a unique code coverage solution for both safety-critical and non safety-critical developers. Project Coverage, which can do both object-level and source-level code coverage, is unique in that it works directly on the executable for the target without instrumenting either the source or the object code and without requiring the actual hardware. How this magic is achieved will be explained in the first part of the presentation.

  The second part will introduce the Open-DO 'as in DO-178B/C) initiative of which Project Coverage is part. The goal of the Open-DO initiative is to provide the foundation of a cooperative and open source

framework for the development of certifiable software. The driving idea is to piggyback on two of the most active and innovative trends that have recently emerged in the software engineering community: (1) Effective collaboration through open source communities, and (2) Empowering development methodologies such as Agile, Lean and eXtreme programming.

- **Ada Programming Language Use in Lockheed Martin – An Update**

  Judith Klein, Lockheed Martin (USA)

  The United States Federal Aviation Administration's mission is to provide a safe, secure, and efficient global aviation system that contributes to national security and the promotion of U.S. aviation. To that end, FAA depends on large, complex, highly available software systems to manage the vast National Airspace System (NAS). Ada is used extensively in the development of said systems. As a system architect on En Route Automation Modernization (ERAM), I will discuss advantages and challenges encountered in the development of ERAM.

  An informal study on the topic of Ada's use in Lockheed Martin will provide the basis for discussion. The kinds of programs/projects using Ada, the size of the Ada development efforts, the estimated life span of the programs/projects, and general comments on vendors supplying products in support of the development effort are aspects in the survey.

Pattern-Based Refactoring Shrinks Maintenance Costs

John S. Harbaugh
The Boeing Company, Seattle, WA, USA; Tel: +1 253 657 5338; email: john.s.harbaugh2@boeing.com

Abstract

Once fielded, software systems enter the maintenance phase of their life cycle. During this phase, changes will be made to improve a system's functionality. An emphasis on functionality alone can lead to a situation where structural qualities assume a lower priority. Software developed under these conditions can, over time, become more difficult, and therefore more costly, to maintain. This paper presents the results of a recent effort by The Boeing Company to improve the data recording and extraction capability of a large Ada 95 Command, Control and Communications (C3) system. It shows how the reliability and structural qualities of the software were improved using the software engineering perspectives of design patterns and refactoring. These improvements led in turn to a high level of maintenance-phase productivity for current and future work.

1 Introduction

The technical details of this effort were influenced by the software engineering topics of "design patterns" popularized by Gamma et al. [1], and from the code maintenance practice known as "refactoring" and described by Fowler [2]. The synthesis of these topics was first discussed by Kerievsky [3].

Both of these approaches work best when applied as first principles. During the course of our work, certain principles kept proving useful to guide our efforts. From *Design Patterns*, two principals kept occurring: One that favors encapsulating those parts of the system that change, and another that favors aggregation over inheritance. From refactoring, we realize the continuing benefits of low coupling and high cohesion for any large computer program.

A third influence on our work was the Ada 95 language itself. Since our work was to be done in Ada 95 and there were relatively few guides or examples of design patterns as realized in Ada 95 (unlike C++ or Java), we needed to study the problem at a fundamental level. The result simultaneously improved the architecture qualities of the software while causing minimal changes to the existing mission computing subprograms. By taking advantage of the uniquely Ada features of hierarchical packages and subtyping, typically no more than minor specification changes were required.

2 Original Design

Our experience for this paper comes from work done on a large (>5MSLOC) Command, Communications, and Control (C3) computing system that was implemented using the Ada 95 computing language [4]. This system controls a variety of sensors, and fuses their information with information from various data links. Crews of trained operators control the overall system using a variety of display consoles. During a mission, data from these sources are recorded as Ada streams to a Mission Data Recorder (MDR) for later extraction and analysis.

Three distinct sources of data are recorded during a mission. Sensor and data link I/O consume the greatest bandwidth, and are recorded from the High-Speed Interface (HSI) in interface-specific binary formats. Data from applications internal to mission computing are another significant contributor to mission data recordings. Application data are recorded as Ada streams of either record types or tagged types. Operator console inputs account for a third distinct source of recording data. All three recording streams are recorded to the in-flight media, then transferred to a mission support center for post-mission extraction and reporting. Following a mission, the mission support center extracts mission data recording into various intermediate formats and distributes them to the end. Figure 1 illustrates the overall data flow between these activities.

The original data extraction program ran as a tightly-coupled process within the mission-computing program, sharing data types and subprograms with mission computing and requiring all of mission computing be operating in order to extract any data. Furthermore, at elaboration each recorded class registered a callback to an associated extraction method with a callback registry. This callback was then used by the data extraction subsystem to extract the mission data recording into the desired form.

Our customer wanted to streamline the processing of post-mission data, and recognized that requiring the entire mission-computing program be running in order to perform any extraction was a significant bottleneck, one that consumed significant computing resources, and required several minutes to start. They desired a solution that would allow data extraction to run separately from the mission computing software, would be a basis for future product development, and could be achievable on a small budget and short schedule. They turned to the Boeing Company for our experience with large C3 systems and Ada development to deliver a comprehensive solution.

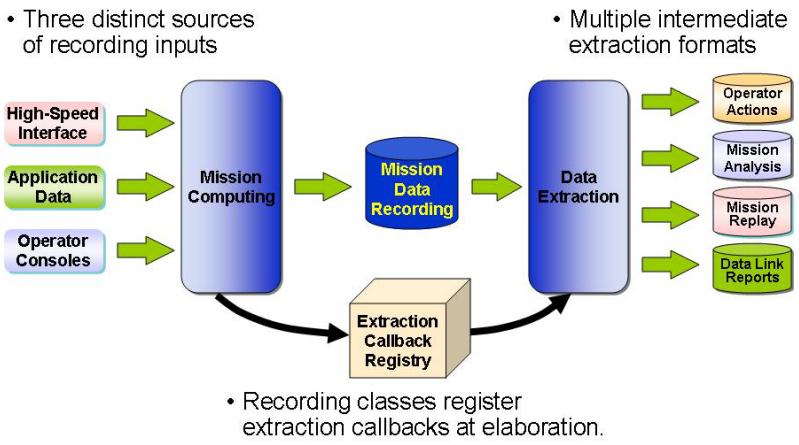

Figure 1 Original Recording & Extraction Data Flow

2.1 Entangling Dependencies

The original data extraction program was designed as a process within the overall mission-computing program. This introduced a web of closure dependencies during compilation. Examples of this coupling included:

- Dependence on mission-computing environment variables established via shell scripts and checked at startup.
- Distributed system management services – All mission-computing processes are controlled by distributed system management.
- Use of CORBA communications – CORBA implementation is tightly tied to system management and mission computing threading architecture.
- Use of distributed event logging.

Even though the data extraction process did not use these features of mission computing, data extraction could not run without them. Thus, the essential problem we faced was not so much functional as it was structural: How to allow the concerns of mission computing to vary independently from those of data extraction, while minimally affecting the existing code base?

The sources of this tightly-coupled structure were twofold. The first was the way in which the recorded data was declared. The original developers followed the common Ada practice of declaring a private type and its primitive operations in a single package specification. Either because of the data type's closure, or because of the closure of the package it which it is declared, entangling dependencies abounded. Since the same data type was used in both mission computing and data extraction, declaring an object of any of these data types would bring in their entire closure, i.e., all of mission computing. In order to de-couple data extraction from the rest of mission computing, the operational dependencies of the objects needed to be separated from those of data extraction, with the data extraction context dependent on a small set of basic data types.

The second was the coupling caused by indirect execution of the class-specific extraction methods via callback. When mission computing elaborated at startup, each recording client registered its externalization methods as callbacks in a common registry. The data-extraction process retrieved these methods as needed, using them to read and convert the recorded data to the externalized form. For data extraction to run without the remainder of mission computing, the externalizations methods needed to be made independent of classes with the aforementioned problem with dependencies.

3 Pattern-Based Solution

3.1 Recordable Type Pattern

Several design patterns, including strategy, decorator, adaptor, and façade, were explored for creating a stand-alone data extraction capability. The key architectural qualities (i.e., variability, extensibility, coupling/cohesion, complexity, efficiency) and cost were evaluated to determine a preferred approach. The conclusion was to implement a strategy pattern using Ada95 to encapsulate the distinct processing needs of the mission computing and data extraction. Figure 2 illustrates our "light weight" implementation of the strategy pattern using Unified Modeling Language (UML) notation.

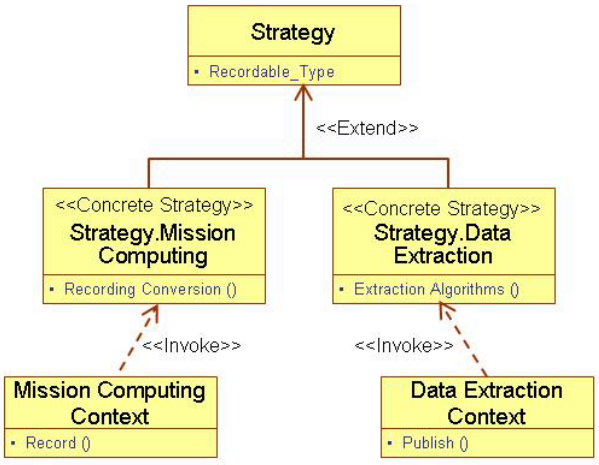

Figure 2 Strategy Pattern Implementation

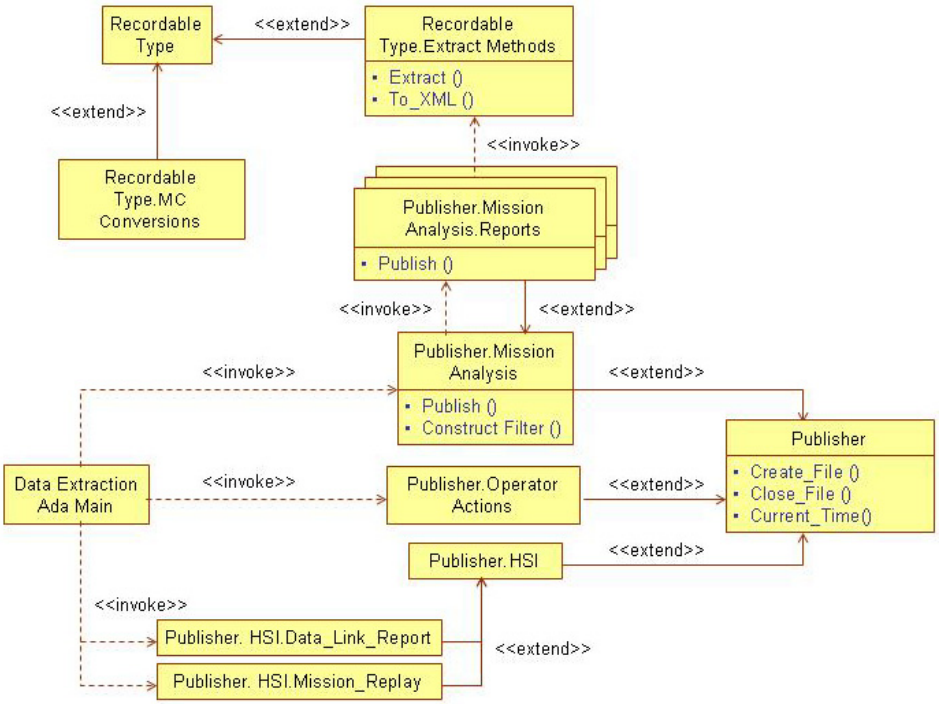

Figure 3 Static Publisher Design

In the context of Ada, we use the solid association arrows to indicate a parent/child relationship, and a dashed association to indicate simple "withing". This convention reflects the stronger form of visibility within package hierarchies. In this implementation, the boxes represent packages and not actual subclasses, as would be expected in a purely object-oriented approach. For this reason, use of inheritance notation would be misleading.

The strategy class is implemented with a new "recordable type" for each application type being recorded. The recordable type is a (potentially) discriminated record type with no significant dependency on other mission computing data types, and which cannot introduce problematic dependencies through closure. This recordable type is declared publicly in a package specification without any primitive operations.

For our system, the strategy was made concrete by extending it with two child packages, one for mission computing conversion operations, and another to support operations specific to data extraction. Each context (i.e., mission computing or data extraction) then "withs in" the desired concrete strategy, getting only the methods (and closure) needed. By segregating contextual code in separate packages, we can ensure that the mission computing program cannot include extraction code and visa versa. If needs change, the pattern easily expands to include new concrete strategies.

In terms of design patterns, hierarchical packages support the concept of aggregation. In *Design Patterns*, the authors recommend favoring aggregation over inheritance. This lightweight implementation takes this advice to the extreme by favoring aggregation to the exclusion of inheritance. We consider this a light-weight implementation because of the lack of inheritance, and because the recordable type is public. Consideration was given to making the recordable type private, with getter and setter methods for each component, but this approach offered no real benefit over a publicly declared type.

We used two approaches to translate mission-computing data to a recordable type. For data in the mission-computing application layer, conversion routines were introduced into the concrete mission computing strategy to convert the domain data representation to its recordable counterpart. This minimizes perturbations to the existing code base, thereby reducing the potential for introducing errors into existing mission computing code

For the High-Speed Interface (HSI), use of conversion routines could have introduced unacceptable processing overhead due to the high data rates involved. Fortunately, these data types were already free of entangling dependencies internally, but still were problematic because of the closure of their declaring packages. For these types, the mission computing type declarations were relocated to a strategy package as a recordable type. The original declarations were then replaced with subtypes (for numeric types and strings) or derived types (for enumeration types) of what was now a recordable type. Because no additional constraints are imposed, the subtype functions as essentially a renaming of the recordable type. Existing references to what are now subtypes require nothing more than recompilation.

Due to the unique syntax of enumeration types, it turned out to be least invasive to the existing code base to redeclare them as derived types rather than subtypes. Changes were limited, typically requiring nothing more than adding a few explicit type conversions. Had we used subtypes instead, every existing literal value would have had to be changed to name the new strategy package.

3.2 Static Publishers

Since data extraction was to become a stand-alone program, a registry of extraction callbacks was no longer needed. Instead, a set of statically invoked "publishers" replaced the callback registry, as illustrated in Figure 3.

Here, a package hierarchy was used to aggregate functionality common to all publishers, then to general groups of publishers, and finally to the individual concrete publishers. As with the recordable types, the package hierarchy for publishers is used to aggregate functionality, not type extension. Each concrete publisher "withs in" recordable extraction methods as needed. In the interest of clarity, Figure 3 shows this relationship only for the mission analysis publishers.

Compared to the previous callback registry, this approach is considered to be a more reliable solution, due to the determinism and compile-time checking of static method invocation.

4 Conclusions

By combining the techniques of design patterns and refactoring, our team of two developers was able to complete a challenging job within an aggressive schedule. Understanding the existing code base and settling on an implementation consumed approximately 50% of the entire effort, with the remainder devoted to code and test. The introduction of the recordable types made it possible to not disturb the existing code base. In fact, all of the existing recording and data extraction code was left intact. Because of its tight coupling and loose cohesion, the effort to remove the existing code and re-qualify the system would have consumed the available budget. In the process of refactoring, we achieved substantial reuse of existing bit-level extraction code while enhancing overall code maintainability with a design that can easily be extended to new publishers in the future.

The benefits of pattern-based refactoring can be realized in any modern, object-oriented language. Each possesses unique properties for encapsulation, name space, inheritance, and threading. Our experience helps confirm the value of design patterns, not as cookie-cutter solutions, but as guides to reasoning about the problem at hand and evaluating potential solutions relative to the implementation language.

The pattern-based design was technically successful and was also successful from the perspective of program management. Due to the modular nature of the strategies and publishers, project scheduling and tracking was straightforward. The consistent "cookie cutter" form of the strategies made it possible to bring in other developers to implement individual recordable types as their schedules permitted, allowing greater utilization of the entire staff.

The improved structure gained by applying pattern-based refactoring allows existing capabilities to be modified, and new ones added with the knowledge that the effort will be localized. Beyond enabling us to meet the cost, schedule and quality goals of the current effort, the time spent improving the code structure has laid the groundwork for future business.

References

[1] Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helms, Johnson, Vlissides, 1995

[2] Refactoring. Improving the Design of Existing Code. Martin Fowler, 1999

[3] Refactoring To Patterns. Joshua Kerievsky, 2004

[4] Ada 95 Reference Manual ISO/IEC 8652:1995(E). S. Tucker Taft and Robert Duff, 1995

Building Cross Language Applications with Ada

Quentin Ochem

AdaCore, 46 rue d'Amsterdam 75009 Paris FRANCE; Tel: +33 (0)1 49 70 66 42; email: ochem@adacore.com

Abstract

This tutorial concerns multi language programming, and in particular multi-language programming with Ada. We'll discuss interfacing solutions with native languages such as C, C++ and languages running on virtual machines such as Java or Python. We'll also consider middleware such as Corba or web services.

Keywords: C, C++, Java, binding, python, CORBA, wsdl, Ada.

1 Introduction

Building complex applications often requires putting together pieces of software or requirements that have never before been made to work together. Thinking of a project with a high integrity kernel written in Ada, using a set of low level libraries and drivers written in C or C++, with a graphical interface done in Java and unit tests driven by Python is not thinking of science-fiction anymore. It's actual concrete and day-to-day work. Unfortunately, having all of these technologies talking to one another is not straightforward and often requires a deep knowledge of both sides of the interface along with extensive manual work.

In this tutorial, we'll first study how to directly interface Ada with native languages, such as C or C++. We'll then look in depth at communications with languages running on virtual machines, such as Java, Python and the .NET framework. Finally, we'll see how Ada can be interfaced with an arbitrary language using a middleware solution, such as SOAP or CORBA. We'll see how communication can be done manually by using low level features and APIs, and how a substantial part of this process can be automated using high level binding generators.

2 Interfacing criteria

There are three main issues to consider when dealing with language interfacing: safety, efficiency and simplicity.

Safety is usually achieved by carrying out as many checks as possible at compile-time. Unfortunately, interfacing often requires designating objects and subprograms by strings only checked at run-time, which often leads to problems which only become apparent at run-time.

Efficiency depends on the way calls are made and data is transmitted. Calls may or may not require indirection; data may or may not require copies, and transfer across a network. Significant performances loss may appear when interfacing through e.g. protocols such as Sockets.

Simplicity is probably the most important criteria of all for programmer. It's about how easy it is to write the interface layer. While it's always possible to do things manually, tool vendors often provide either stub generators, or even better, direct binding generators from one language specification to the other.

3 Ada and C

Interfacing Ada and C is probably the most common interfacing scenario. Having said that, it carries a lot of potential mistakes and should be done with great care. Programmers may need to be aware of low level details on the memory layout and calls convention, as Ada objects & subprograms are not compatible with C by default.

Programmers are advised to keep things as simple as possible. Advanced C structures such as unions, arrays or strings can still be bound but need additional knowledge on the compilers behaviour.

4 C++ and binding generators

While writing manually C interfacing can be tedious, using an automatic binding generator is probably a comfortable solution. Recent version of gcc distributed with GNAT GPL and GNAT Pro include a new switch automatically generating an Ada specification out of a C or C++ header file.

In addition to the C constructions, the binding generator also supports C++ classes, and maps them into Ada tagged types. Such types can then be derived in Ada, and primitives can be overridden, leading to cross-language classes and cross language dispatching.

5 GNATCOLL.Scripts and Python

GNATCOLL is a collection of components distributed with GNAT, and contains amongst other things a standard interface to scripting languages. The default backend provided works for Python, but any scripting language with reflective capability would fit into the framework. A set of functionality exporting Ada subprograms to a scripting environment, or calling directly scripting code is provided. This tool has been extensively used in the development of GPS – the GNAT Programming Studio – for providing an interface for extensibility and as a mean to drive tests.

6 Java and GNAT-AJIS

Java and native languages such as C are most often interfaced using the JNI – the Java Native Interface. Unfortunately, this layer is extremely tedious to use, and provides almost no type safety.

In order to workaround these problems, GNAT now comes with a binding generator from Ada to Java, giving a callable Java interface generated from an Ada specification. This generator supports a wide range of Ada features, such as subprograms, global variables, access to subprograms, exceptions... Ada tagged types are mapped to Java classes, which can then be extended in Java. As for the C++ case, the resulting classes are cross-language, and can be used in a cross-language dispatching call.

This tool has been extensively used in the GNATbench Eclipse Plugin, in order to retrieve code that have been previously written in Ada for the GNAT Programming Studio.

7 The JVM and .Net

Both the JVM and the .Net bytecodes embed a specification description. This specification can then be used to generate a direct binding. This is the purpose of e.g. the jvm2ada tool, taking a Java bytecode .class file, and generating an Ada specification for the GNAT for the JVM compiler, or the cil2ada tool doing the same for the .Net platform.

This feature is particularly impressive on the .Net platform. There are a huge number of languages targeting .Net, each of them generating this intermediate representation in the bytecode. This makes it possible to interface practically any languages with any other one, provided that one binding generator has been provided from the common representation to the target language.

8 CORBA

While it has been originally designed for distributed programming, CORBA has been used in a number of cases to solve multi-language programming issues. It provides an intermediate specification representation, the IDL, coming with stub generator for most languages.

IDL generators can be used to create a callable interface, or stubs for the implementation code. Since this language is the common denominator of all, programmes don't have to care about the interface between two or more particular languages anymore, but only about the way of interfacing to that common representation. However, the communication layer is implemented on top of sockets, and the performance loss can easily be a show stopper when many data exchanges are involved.

9 Web languages

Web languages traditionally lie within the boundaries of scripting languages, such as Perl, PHP or JavaScript. However, the notion of web application is emerging, with a whole new set of challenges going much further than websites in terms of application architecture and data computation. For these applications, using structured and constrained language such as Ada is starting to make sense, hence the question regarding the interfacing with the existing code.

Server-side applications can be interfaced with regular techniques which have been explained so far. However, a common way of creating multiple language – or peer – applications is to rely on web services. These services somehow act in the same way to CORBA, and comes with a specification language conceptually close to the IDL : WSDL.

Client side computation currently lacks solutions – JavaScript seems to be the only portable technology available natively on all browsers. Google decided to bet on that fact, and provided a Java to JavaScript compiler as part of its GWT toolset. An Ada to JavaScript compiler could be envisioned in a similar manner. The question of directly interfacing JavaScript code with Ada raises a whole new set of interesting questions, JavaScript begin extremely relaxed in terms of typing.

Conclusion

Multi-language programming certainly carries doubt and raises a huge number of questions, which are becoming more and more unavoidable these days. Fortunately, tool vendors have invested a lot of time and effort into making these things easier, and can now provide users with advise, support and tools, now making this reasonable to do in an industrial and challenging context.

Execution Time: Analysis, Verification and Optimization for Reliable Systems

Robert I. Davis and Ian Broster

Rapita Systems Ltd, York Science Park, Heslington, York, YO10 5DG, UK; Tel: +44 1904 567747; email: rob.davis@rapitasystems.com, ianb@rapitasystems.com

Abstract

Verifying and optimizing real-time software to meet timing requirements can be a costly and time consuming task. To achieve reliable timing behaviour, it is necessary to focus attention on the worst case. In this paper, we discuss how detailed execution time measurements can be used to provide an accurate worst-case execution time estimate for embedded real-time software, and also to identify those subprograms that contribute most to the worst-case execution time. We describe a strategy for targeted optimization of these worst-case hotspots that provides a cost effective way of ensuring that time constraints are met, as well as creating headroom for additional functionality, without the need for costly hardware upgrades.

Keywords: execution time, worst-case execution time, WCET, timing analysis, optimization.

1 Introduction

Software execution time can be identified as a fundamental cause of correct or incorrect timing behaviour in embedded real-time systems. Depending on the nature of the system and its requirements, there may be a more or less stringent need to show that the system will always meet its timing requirements. For example, the developers of safety critical systems, such as flight control systems, must be able to show to a very high degree of confidence that the software will always execute within its time constraints. In other systems, incorrect timing behaviour can have a major impact on the performance and reliability of the system and the customer's perception of product quality.

The fundamental reason why it is difficult to obtain reliable software timing behaviour is that the source code does not provide information about execution time. Contrast this with subprogramal behaviour, where, to a large extent, the programmer is able to reason precisely about the software behaviour by reviewing the source code. The execution time of source code, on the other hand, depends not only on the code, but also on the compiler, the target hardware, and interactions with other parts of the system.

To obtain confidence in the timing behaviour of a real-time system, engineers typically measure the end-to-end execution time of major software components such as partitions and tasks during system tests. These measurements reveal how often, if at all, execution time budgets are exceeded.

Recording a series of end-to-end execution time measurements allows engineers to see the distribution of execution times for each major software component and also the high-water mark: the longest execution time that was actually observed. This is often used as an indicator of the worst-case execution time, and gives some degree of confidence in the timing behaviour of the system.

In this paper, we examine the process of software execution time analysis, verification, and optimization to achieve within-budget execution times. In Section 2, we discuss the limitations of high-water mark measurements and outline why it is important to obtain more detailed timing information. In Section 3, we present a proven, effective and efficient strategy aimed at resolving the problem of execution time budget overruns. In Section 4 we outline the results of a study where this strategy was applied.

2 Why detailed timing information is important

There is a fundamental issue inherent in the simple end-to-end measurements used to obtain high-water mark execution times. The high-water marks may not reflect the longest time that the code could take to execute. This typically happens when the longest path through the code has not been exercised by the tests.

While code coverage tools can be used to make simple checks that the available tests cover all of the code, for example statement, condition and decision coverage, it is possible, and in fact is often the case, that the longest path through the code has not been executed by the tests. This is due to different sections of code taking their longest execution times at different times, and for different values of input data.

For non-trivial code, it is very hard to devise tests that are certain to drive the code down its longest path. This is because the number of distinct paths increases exponentially with the number of decision points in the code. Further, when there are loops in the code, it is not in general possible to tell from end-to-end measurements, whether the loops have executed for their maximum possible number of iterations.

End-to-end execution time measurements are useful in obtaining basic confidence in system timing behaviour; however, they are not a reliable indicator that execution

time budgets will always be met once the system is deployed. Over-reliance on these end-to-end measurements can result in operational problems with deployed systems that can eventually and painstakingly be traced back to budget overruns. All too often, these are the hardest and most expensive problems to solve due to the intermittent nature of the faults.

Further, as software is developed, and more features are added, for example during software upgrades, the high-water mark execution times may show that the software takes too long to execute; however, these measurements provide no information about which parts of the code contribute most to the overall execution time, and so they offer no indication as to what code to optimize. This is a significant problem, as optimizing code that is not on the worst-case path is worse than a waste of time; it does not address the timing issues, and wholesale attempts to re-write large volumes of code to be more efficient are extremely costly and may introduce bugs into the software.

An effective way of obtaining much more detailed timing information is to add lightweight instrumentation points at each decision point in the code. Whenever it is executed, each instrumentation point outputs its identifier, which can be time stamped and recorded by a suitable data capture device [1]; thus running a series of tests on the instrumented system results in the creation of a timing trace. By combining timing measurements from the trace data with structural information obtained from analysis of the code, it is possible to determine a wealth of information about the timing behaviour of the software, including:

- The worst-case execution time of the software, even if the worst-case path has not actually been executed during testing.
- How many times each subprogram and sub-path was executed during testing. If some sub-paths have not been covered, then there are gaps in testing which should be addressed.
- The maximum number of iterations for each loop. These can be compared with predictions to see if the expected maximum number of iteration is seen, or even exceeded in practice.
- Which lines of code are on the worst-case path, and equally important, which ones aren't and so do not rate as candidates for optimization.
- How much each subprogram and each sub-path contributes to the worst-case execution time. Identifying the sections of code with high contributions to the worst-case execution time, so called *worst-case hotspots*, is an excellent way of finding the best candidates for optimization when the overall execution time needs to be reduced.
- How the end-to-end execution times of each subprogram vary over the different tests, and hence show the correlation between particular test cases and long execution times.

For commercial scale applications, it would be possible although extremely laborious to instrument programs by hand; however, the volume of trace data typically produced would make manual attempts to combine trace data with program structural information infeasible. Fortunately, the tasks of program instrumentation, trace processing, combining trace data with program structural information, data mining, and presentation are all amenable to automation. Automated execution time and performance measurement tools, such as the RapiTime toolset [2] from Rapita Systems Ltd. provide an integrated solution to these problems.

The availability of detailed timing information has the following benefits:

- It enables engineers to take a systematic and scientific approach to obtaining confidence in the timing behaviour of the system. As opposed to spending a great deal of time and effort trying to track down intermittent timing bugs revealed post deployment.
- Detailed information about the contribution of subprograms and blocks of code to the worst-case execution time, enables *worst-case hotspots* to be identified that form potential candidates for optimization.
- "What-if?" analysis quantifies the maximum performance gains obtainable by optimizing selected software components. Together, with hotspot analysis, this ensures that optimization effort is only applied where it will have the maximum benefit in terms of reducing the overall execution time, and ensuring correct timing behaviour of the system.

3 Optimization strategy

When detailed execution time measurements are taken during extensive testing, they can highlight problems where certain software components overrun or have the potential to overrun their budgets. Such problems can occur during initial system development, or as a result of adding new functionality as part of a mid-life upgrade. When faced with the problem of a software component that overruns its execution time budget, it is essential that a systematic and scientific approach is taken to resolving the problem. The following sections describe a proven, effective and efficient strategy for dealing with the problem of execution time budget overruns:

3.1 Execution time budget re-allocation

The simplest and most cost effective solution to execution time budget overruns is to increase the execution time budget, at the expense of some other software component that has been shown not to require its entire execution time budget. For this re-allocation to be effective, it is of course important to have accurate timing information about all of the software components in the system, otherwise the timing issue is simply shifted from one component to another.

If budget re-allocation alone cannot be used to solve the problem, then the next course of action is normally software optimization. We note that in a system with many components, for example multiple partitions and tasks, not all of the execution time reduction required need come from the overrunning component. A mixture of budget re-allocation and optimization could be more effective, as assessing a number of software components provides more scope for finding subprograms which contribute significantly to the worst-case execution time, and yet can be simply and easily optimized.

3.2 Selecting candidates for optimization

It is important to be clear about the goal of optimization. To achieve reliable timing behaviour, the worst-case execution time of a software component must be within the specified execution time budget; hence we are interested in optimizing the worst-case execution time, not the average-case execution time. Optimizing for the worst case is different from optimizing for the average case. It requires a focus on code on the worst-case path, which is often not the same as the code that executes most frequently.

Many well known optimizations, while excellent for improving average-case performance actually make the worst-case execution time worse. For example, early exits from loops can reduce the average number of iterations of a loop, reducing the average-case execution time. By contrast, when the worst case is considered, adding a conditional test for an early exit from a loop does not change the maximum number of iterations that the loop can take; however, it does add additional code that is executed on every iteration of the loop increasing the worst-case execution time.

Optimization is a compromise between several different factors, in particular: execution time, code and memory space, readability, maintainability and effort. For example, some optimizations may lead to code structures that are very hard to maintain but result in a significant reduction in execution time. The key to an effective optimization strategy is to prioritize those optimizations where the minimum effort, and the minimum amount of compromise in other factors, is required to gain the maximum benefit in terms of execution time reduction. In this respect, access to detailed timing information is necessary to manage the optimization process effectively.

Firstly, it is important to identify, those subprograms or sections of code that contribute the most to the overall worst-case execution time. These worst-case hotspots represent potential candidates for optimization.

Worst-case hotspot analysis of complex applications shows a common trend, typical we see that:

- Most subprograms are not actually on the worst-case path, and so contribute nothing to the worst-case execution time. Optimization of these subprograms would not reduce the worst-case execution time at all.
- Many subprograms contribute a small amount to the worst-case execution time and so do not represent good candidates for optimization. Expending effort reducing the execution time of a subprogram that contributes less than 1% to the total is unlikely to be worthwhile.
- A small number of subprograms contribute a large fraction of the overall worst-case execution time and are therefore potential candidates for optimization.

Using worst-case hotspot analysis, engineers can easily identify the relatively small number of subprograms where optimization could potentially have a large impact on the overall worst-case execution time. However, before reviewing the source code of the top 10 to 20 candidates for optimization, it is important to ask the following questions:

"What if the execution time of this subprogram were reduced by 50%, 80% or even 100%, what would the effect on the overall worst-case execution time be?"

Asking this question is important, because even though a subprogram may be a worst-case hotspot, its optimization may not necessarily lead to a significant reduction in the overall worst-case execution time if by optimizing that code, the worst-case path switches to another path. For example, consider the code fragment below:

```
If some_condition then
   A; -- in worst-case path. Takes 10 ms
else
   B; -- not in worst-case path. Takes 5ms
end if;
```

In this example, reducing the execution time of subprogram A by more than 5 ms, switches the worst-case path to subprogram B, therefore both subprograms A and B need to be optimized together to reduce the worst-case execution time further.

Detailed evidence indicating where optimization will have the maximum benefit, and the extent to which optimization can be expected to reduce the worst-case execution time, enables effective management of the optimization process. Typically, a small number of candidate subprograms are selected for optimization. The source code for these subprograms is then reviewed and prototype optimizations implemented.

After completing the prototype optimizations it is necessary to determine what was actually gained. This can be achieved by repeating the timing measurement and analysis process to quantify the reduction in the overall worst-case execution time. Once this second set of results has been obtained, then a decision can be made as to which optimizations to keep. If the gain is not significant compared to the compromise in other factors, such as maintainability, then a prototype optimization may be discarded.

Finally, if the results show that the optimizations have resulted in the execution time budget being met, then the prototyped optimizations can be included in the main code base. If the execution time budget is still exceeded, then further candidates for optimization can be examined, or

budget re-allocation and optimization of another sub-system considered.

To summarise, a proven strategy for resolving execution time budget overruns is as follows:

1. Obtain detailed timing information via measurement of on-target execution times, and analysis of the code structure.
2. Re-allocate execution time budgets where possible.
3. Select candidates for optimization based on their contribution to the overall worst-case execution time.
4. Quantifying the maximum potential gains by determining what the impact on the overall worst-case execution time would be if the execution time of each candidate were substantially reduced.
5. Prototype optimizations for the best candidates, i.e. those with the largest potential gains.
6. Quantify the worst-case execution time reductions achieved via timing measurements on the target.
7. Adopt the most effective optimizations into the development code.

4 Hawk Mission Computer: Operational Flight Program

This section describes how the above strategy was applied in a study performed by engineers from Rapita Systems Ltd. and BAE Systems, aimed at reducing the worst-case execution time of software components in the Operational Flight Program of the BAE Systems' Hawk Mission Computer[1]. For a full description of this study see [3].

The Operational Flight Program is written in Ada and consists of hundreds of thousands of lines of code divided into 25 partitions, themselves divided into tasks, executed in a cyclic schedule. This system was running close to capacity, in terms of available execution time. In order to provide capacity for new functionality, a study was conducted to identify optimization opportunities that would reduce the worst-case execution time of the system by at least 10%; thus avoiding the need for an expensive hardware upgrade.

Previous efforts at understanding the timing behaviour of the system had been based on determining the execution time of each partition via high-water marks measured on the target microprocessor. A typical situation was that painstaking optimization of a subprogram would result in unit tests showing a significant reduction in execution time while making little or no impact on the overall high-water mark. In contrast, simpler optimizations could sometimes have a significant impact, reducing the high-water mark readings. This occurred when, in the first case, the code was not actually on the worst-case path, and in the second case, when the subprogram was both on the worst-case path and called a large number of times on that path.

[1] Hawk is a fast jet trainer, famously flown by the Red Arrows display team.

In all, 5 out of 25 software partitions were analysed in the study, amounting to over 100,000 lines of Ada code. Three of the partitions, A, B and C were comprehensively analysed, with improvements and targets for optimization selected on the basis of the information provided using the RapiTime toolset. Optimizations were prototyped for these partitions and the performance analysis re-run to quantify the improvements obtained.

The detailed timing information provided by RapiTime showed that 1.2% of the code contributed more than 29% of the overall worst-case execution time. These blocks of code were obvious targets for optimization. A detailed study of some 1250 lines of code identified specific targets for optimization and hence opportunities for execution time reduction. The best candidates were prototyped and implemented and the new system analysed to verify the effectiveness of the changes. The optimized partitions had an execution time that was over 23% smaller than before, creating headroom for additional functionality without the need for costly hardware upgrades.

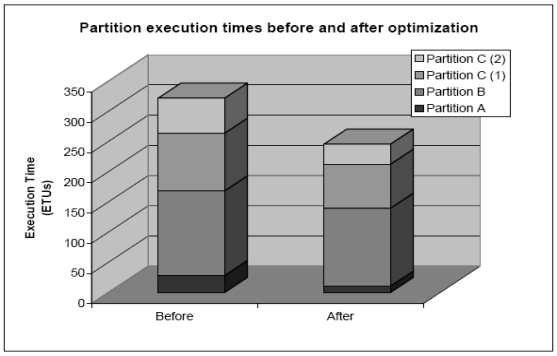

Figure 1: Reduction in worst-case execution time achieved using RapiTime

5 Summary and conclusions

This paper discussed how a focus on worst-case execution times is important in achieving reliable system timing behaviour. The limitations of high-water mark measurements were discussed, and the need for more detailed timing information explained. The key contribution of this paper is the description of a strategy for managing and resolving the problem of execution time budget overruns via a focus on worst-case execution times. This strategy was proven to be both efficient and effective in a study of an Avionics system implemented in Ada (the Operational Flight Program of the BAE Systems' Hawk Mission Computer).

References

[1] "RTBx1200 Series Trace Data Loggers" Rapita Systems Ltd. Data sheet available from www.rapitasystems.com.

[2] "RapiTime White Paper" 12th June 2008, available from www.rapitasystems.com.

[3] G. Bernat, R.I. Davis, N. Merriam, J. Tuffen, A. Gardner, M. Bennett, D. Armstrong. "Identifying Opportunities for Worst-case Execution Time Reduction in an Avionics System". Ada User Journal pp. 189-194, Volume 28, Number 3, Sept. 2007

Software Fault Tolerance

Patrick Rogers

AdaCore, 207 Charleston St., Friendswood, Texas, USA 77546; email: rogers@adacore.com

Abstract

The following is a summary of the tutorial by that name provided at the Conference on Reliable Software Technologies, Ada Europe 2009. Brest, France. 8-12 June 2009.

Keywords: software, fault tolerance, error recovery, Ada.

1 Introduction

Today's extremely demanding applications are made possible by the flexibility and power of digital computer technology. For example, advanced fighter aircraft are made inherently unstable to produce extreme manoeuvrability. Only the speed and flexibility of computer technology make it possible to control such aircraft in flight; a human pilot can not do so.

Unfortunately, errors in large complex systems appear unavoidable with current technology, even given stringent development and testing procedures. Complexity is such that full testing is not feasible and complete proofs of correctness are at best inherently limited by the potential for specification faults.

Tolerating software imperfections is very difficult, and successful approaches used to tolerate hardware faults do not necessarily apply. Typical fault tolerant systems design techniques can not cope with the complexity and scale of current and future applications. Their inadequacy stems from their assumptions: correct algorithm design, knowledge of all possible failure modes, complete knowledge of internal interactions of the components, and complete knowledge of external interactions with the environment. In particular, although the components individually may be well understood, at the point of component and subsystem interaction overall complexity comes into play, precluding anticipation of all possible faults. The problem is, therefore, one of handling *unanticipated* faults.

1.1 Faults, Errors, and Failures

The concepts of fault, error, and failure are directly linked in a causal relationship: faults lead to errors, which ultimately lead to failures. A fault is a physical defect or flaw within a hardware or software component. This is essentially the definition found in a typical dictionary. An error is the manifestation of a fault: a deviation from accuracy or correctness in state. A failure is an externally observable event representing a deviation from the authoritative service specification.

1.2 Causes of Faults

Specification mistakes are considered the cause of the majority of safety mishaps. Specification mistakes include architectural flaws as well as hardware and software design specification errors

Implementation mistakes are the intuitive cause of faults. These mistakes include poor design and construction, poor component selection, and poor software coding.

1.3 Characteristics of Faults

'Transient' faults are temporary. They are caused by circumstances or events that cannot be recreated in a controllable way, such as gamma rays that flip a bit in memory, or highly unusual combinations of events.

'Intermittent' faults enter the system, stay active for a while and then disappear, only to return again.

'Permanent' faults are completely repeatable and (as the name indicates) always cause an associated failure (using the standard causal definition).

1.4 Failure Hypothesis

The chosen system failure model is a critical design decision because it drives the use of redundancy for tolerating faults. This assumption of a failure model is called the 'failure hypothesis'. For a given failure hypothesis, the number of redundant components required can be calculated. The choice of failure hypothesis is one of the central decisions about the form of fault tolerance to be used. Architectural decisions are made on the basis of the number and kinds of faults to be tolerated. In all cases, fault tolerance mechanisms are built to handle the number of faults required, and no more: if more faults are experienced than expected, they will not be handled. As will be seen, some architectures directly detect when the number of tolerable faults is exceeded; others require additional facilities.

2 Dealing With Faults

There are two ways to deal with faults: fault prevention and fault tolerance. Fault intolerance aims to *prevent the existence* of faults. Fault tolerance is intended to *handle* faults when they occur in an executing system.

2.1 Fault Prevention

There are two means of fault prevention: avoiding their introduction during production, and removing them before deployment. In both cases faults are dealt with prior to execution. 'Fault avoidance' is a design activity that attempts to prevent faults from being introduced into the deployed system. 'Design' is intended to include all

production phases of the life cycle, including requirements definitions, design methods, design reviews, programming techniques, testing, and other quality control measures.

'Fault removal' is a design (i.e., implementation) activity focused upon testing. Testing is inherently limited by the inability to test under completely realistic conditions, by the potential for specification errors, and, of course, by the fact that testing can only show the presence of errors, not their absence.

2.2 Fault Tolerance

In contrast to fault avoidance, fault tolerance schemes consider faults inevitable and deal with them *after* deployment. As a run-time activity, therefore, fault tolerance may be defined as the ability of a system to continue to perform in the presence of faults.

If faulty software or hardware components are to be dealt with during execution, some additional resource is necessary. Therefore, fault tolerance is based on one of several forms of redundancy.

3 Software Redundancy Mechanisms

There are two general approaches to achieving fault tolerance: a 'static' architecture and a 'dynamic' architecture. In a static architecture, a given function is executed by several computing resources, and a correct result is selected from one of the outputs produced. The term 'static' is applied to this approach because the result of the adjudication of the outputs does not affect the architecture; subsequent provision of a result is achieved using the same computing resources (in the archetypal usage).

In a dynamic fault tolerance architecture, a given function is executed by a single computing resource. If a result cannot be produced, alternate resources are used to compute the result. The term 'dynamic' is applied because failure to produce a result directly affects the architecture: redundant resources detect faults and alter the architecture accordingly, such that subsequent invocation does not use the same (faulty) computing resources.

3.1 Error Recovery

Error (fault) recovery is the central component of dynamic fault tolerance strategies: it must transform a faulty system into one with a valid, perhaps degraded, state. Two approaches to fault recovery have been identified: backward error recovery and forward error recovery.

Backward error recovery mechanisms attempt to simulate the reversal of time to a point at which the system state was error-free. They do so by saving state when it is assumed to be valid, and then restoring that state as necessary. The act of saving state is called 'taking a checkpoint'.

Forward error recovery mechanisms attempt to make selective changes to an erroneous state, to move to a new, error-free state. Most modern programming languages provide direct support for forward error recovery via exceptions. As a form of forward error recovery, exceptions are not applicable to design faults, since such faults are unanticipated.

3.2 Software Mechanisms for Tolerating Software Design Faults

Continuity of service in the presence of software faults requires at least one additional component capable of meeting the functional specification. Simple replication, however, as applied for hardware fault tolerance, would simply replicate the faulty software.

To avoid the occurrence of the same fault, and on the assumption that software design faults are indeed permanent, this additional component must be implemented differently from the one that failed (although it will be designed to the same specification). *Software diversity* is, therefore, the underlying principle of the systematic mechanisms described in the following sections. Two approaches to diversity have been identified: 'design' diversity, and 'data' diversity. By far, design diversity is the most widely explored and used. In that approach, completely separate, distinct designs and implementations are used, with the expectation that the differences will result in different faults being exhibited (if any).

3.2.1 N-Version Programming

N-Version Programming (NVP) is the software instance of the N-Modular Redundancy (NMR) approach used to handle hardware faults. In the NVP approach, *N* separate, complete variants are developed, and each is executed in parallel whenever the component is invoked. The *N* results are compared at *cross-check* points to detect faults, usually using a majority-based voting adjudicator. Faulty variants are detected by their differences from the other variants' outputs, and are masked out by the adjudicator. As in NMR, the value of *N* is chosen so that the required number of faults may be tolerated. The value of *N* is selected such that the number of faults to be tolerated is less than the majority represented by *N*.

3.2.2 Recovery Blocks

Recovery blocks are the classic example of dynamic redundancy, in that reconfiguration by selection of an alternate variant does not occur until a fault is detected.

In the recovery block mechanism, complete stand-alone programs are not implemented. Rather, for a given unit of functionality, a primary variant is implemented as a module, instead of a complete program; design diversity is incorporated via alternate modules. The primary module is the only one executed under normal conditions. Only when an error is detected by application of an acceptance test do alternative variants execute, and even then only one executes at a time. Alternatives are successively called, and the acceptance test successively applied against the results, until either a satisfactory result is obtained or no remaining alternate variants are available (in which case the failure is propagated). Hypothetical syntax for expressing recovery blocks is shown below:

```
ensure acceptance_test by primary_variant
  else by variant_2
  else by variant_3
  …
  else by variant_N
  else error
```

Central to the concept is that the current, valid state is saved prior to execution of the primary module. This saved state is successively restored prior to the execution of each of the alternative modules whenever the acceptance test fails. Therefore, recovery blocks use backward error recovery.

3.2.3 Error Recovery in Concurrent Applications

The facilities examined for use *within* a process are not applicable by themselves in co-operative concurrent programming environments. They are limited because errors in co-operating processes can propagate amongst the group, since, by definition, such processes communicate. Error recovery in one process must, therefore, involve all that have communicated with that process because their individual states are mutually dependent. Recovery blocks et cetera do not manage recovery activities beyond that of a single thread of control.

Atomic Actions

Atomic Actions are a means of fault propagation control and assessment. They achieve containment by the enforcement of semantics requiring their effects to be both indivisible and instantaneous when viewed by processes not involved in the action. Specifically, no data flow between application processes during an atomic action unless they are participants in that action and partial execution of an action cannot be observed by non-participating processes. The Ada extended rendezvous is an atomic action involving only two processes (not considering nesting). More generally, atomic actions can be built to handle as many processes as necessary.

Conversations

Conversations are a structural combination of atomic actions with backward error recovery (in the form of recovery blocks) for co-operating processes. Conversations ensure a consistent recovery line because each process takes a checkpoint upon entrance to the conversation. Whenever one of the processes involved suffers a fault (i.e., does not pass the local acceptance test) *all* processes roll-back to the common checkpoint, ensuring consistency of the processes' recovery. Furthermore, conversations form a synchronisation boundary to ensure consistency, in that all processes leave the conversation together with a common view of the activities within.

4 Sample Concrete Implementations

Backward error recovery can be supported via reusable components because it is independent of the application. In this part of the tutorial we examined reusable Ada 2005 implementations of backward error recovery in the form of state management, recovery blocks, and conversations.

5 Concluding Remarks

After reviewing the goals of the tutorial and the tutorial concluded with a discussion of whether design diversity is still considered a viable technique, with a number of references provided for both sides of the debate.

6 Suggested Reading

M. Lyu, Ed. Software Fault Tolerance, in Trends In Software, vol. 3, Chichester: John Wiley & Sons, 1995.

L. Pullum, Software Fault Tolerance Techniques & Implementation: Artech House, Inc., 2001.

A. Burns and A. J. Wellings, Real-Time Systems and Programming Languages, 3rd. ed: Addison-Wesley, 2001.

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/.

Gem #39: Efficient Stream I/O for Array Types

Pat Rogers, AdaCore

Date: 9 June 2008

Abstract: Reading and writing values from/to streams is easy with Ada's "stream attributes" but for some array types the default attribute implementations could be made more efficient. In this Gem we show how the user can define these more efficient implementations.

Let's get started…

Ada has the notion of "streams" that are much like those of other languages: sequences of elements comprising values of arbitrary, possibly different, types. Placing a value into a stream is easy using the language-defined "stream attributes". The programmer simply calls the type-specific attribute routine and specifies the stream and the value. For example, to place an Integer value V into a stream S, one could write the following:

```
Integer'Write (S, V);
```

Strictly speaking, S is not a stream but, rather, an access value designating a stream. The Integer'Write routine will convert the value of V into an array of "stream elements" – essentially an array of storage elements – and then put them into the stream designated by S. Actually, placing the bytes into the stream is accomplished by dynamically dispatching to a procedure specific to the stream representation.

Although this discussion is couched in terms of placing values into streams, you should understand that reading values from streams is very similar to writing them and that the same efficiency issue and solution apply.

For composite types, such as array or record types, each component value is individually written to the stream using the approach described above. Consider an array type "A" specifying Integer as the component type. The default version of A'Write will call Integer'Write for each component. Thus, each Integer value is converted to the array of storage elements and written to the stream. This component-driven behavior is necessary because programmers can define their own versions of the stream attributes, and naturally will expect them to be called even when the types in question are used as component types within enclosing array or record types.

But suppose the array type is structurally just a sequence of contiguous bytes, and the component type does not have a user-defined stream attribute defined. In that case, calling the component-specific attribute for each array component is unnecessary and inefficient.

For example, suppose you are working with Military-Standard 1553B for communicating application values between remote devices. Ultimately, Mil-Std-1553B sends and receives 32-word buffers, where each word is an unsigned 16-bit value. Suppose as well that you want to write and read these buffers to and from streams. We can override the stream attributes so that a whole buffer value is written directly to the stream instead of writing it one buffer component at a time.

The buffer type could be declared as follows:

```
type Buffer is array (1..32) of Interfaces.Unsigned_16;
```

We can then override the stream attributes for type Buffer.

First we declare the routines:

```
procedure Read_Buffer
  (Stream : not null access
       Ada.Streams.Root_Stream_Type'Class;
   Item   : out Buffer);

procedure Write_Buffer
  (Stream : not null access
       Ada.Streams.Root_Stream_Type'Class;
   Item   : in Buffer);
```

All such stream attributes have the same formal parameter types, i.e., an access parameter designating the class-wide root stream type defined by the language, and the type to be written to, or read from, that stream.

We then "tie" the routines to the stream attributes for type Buffer, thereby overriding the default versions:

```
for Buffer'Read use Read_Buffer;
for Buffer'Write use Write_Buffer;
```

The language-defined root stream type and array element type are declared in package Ada.Streams:

```
package Ada.Streams is

  type Root_Stream_Type is abstract
     tagged limited private;

  type Stream_Element is mod 2 ** Standard'Storage_Unit;
  type Stream_Element_Offset is range
    -(2 ** (Standard'Address_Size - 1)) ..
    +(2 ** (Standard'Address_Size - 1)) - 1;
  …
  type Stream_Element_Array is
    array (Stream_Element_Offset range <>) of
       aliased Stream_Element;

  procedure Read
    (Stream : in out Root_Stream_Type;
     Item   : out Stream_Element_Array;
     Last   : out Stream_Element_Offset)
  is abstract;

  procedure Write
    (Stream : in out Root_Stream_Type;
```

```
    Item   : Stream_Element_Array)
  is abstract;
  …
end Ada.Streams;
```

The user-defined Read_Buffer and Write_Buffer routines will call these stream-oriented Read and Write procedures (via dynamic dispatching) once for the entire Buffer array value, instead of calling them once per array component. Both routines are very similar, so we will omit the body of of Read_Buffer for the sake of brevity and show just the implementation of Write_Buffer:

```
procedure Write_Buffer
  (Stream : not null access
            Ada.Streams.Root_Stream_Type'Class;
   Item   : in Buffer)
is
  Item_Size : constant Stream_Element_Offset :=
           Buffer'Object_Size / Stream_Element'Size;

  type SEA_Pointer is
    access all Stream_Element_Array (1 .. Item_Size);

  function As_SEA_Pointer is
    new Ada.Unchecked_Conversion (
           System.Address, SEA_Pointer);
begin
  Ada.Streams.Write (
           Stream.all,
           As_SEA_Pointer (Item'Address).all);
end Write_Buffer;
```

In the above, we cannot simply convert the value of Item, of array type Buffer, to a value of type Stream_Element_Array, so we work with pointers instead. We define an access type designating a Stream_Element_Array that is the exact size, in terms of Stream_Elements, of the incoming Buffer value. Note the use of the Buffer'Object_Size attribute in that computation. That attribute gives us the size of objects of the type Buffer, a wise approach since in general the size of a type may not equal the size of objects of that type. We can then use unchecked conversion to convert the address of the formal parameter Item to this access type. Dereferencing that converted access value (via .all) gives us a value of type Stream_Element_Array that we can pass to the call to Ada.Streams.Write.

Thus we avoid processing each component of type Buffer, instead writing the entire Buffer value at once. That's a much more efficient approach. As we said earlier, reading values from streams is analogous to writing values to them and only differs in obvious, minor ways. That is true for using the default stream attributes as well as in the implementation of Read_Buffer.

Gem #50: Overload Resolution

Bob Duff, AdaCore

Date: 27 October 2008

Abstract: This Gem discusses some language-design issues related to overload resolution.

Let's get started…

Ada allows overloading of subprograms, which means that two or more subprogram declarations with the same name can be visible at the same place. Here, "name" can refer to operator symbols, like "+". Ada also allows overloading of various other notations, such as literals and aggregates.

In most languages that support overloading, overload resolution is done "bottom up" — that is, information flows from inner constructs to outer constructs. (As usual, computer folks draw their trees upside-down, with the root at the top.) For example, if we have two procedures Print:

```
procedure Print (S : Sequence);
procedure Print (S : Set);
X : Sequence;
…
Print (X);
```

the type of X determines which Print is meant in the call.

Ada is unusual in that it supports top-down overload resolution as well:

```
function Empty return Sequence;
procedure Print_Sequence (S : Sequence);
function Empty return Set;
procedure Print_Set (S : Set);
…
Print_Sequence (Empty);
```

The type of the formal parameter S of Print_Sequence determines which Empty is meant in the call. In C++, for example, the equivalent of the "Print (X)" example would resolve, but the "Print_Sequence (Empty)" would be illegal, because C++ does not use top-down information.

If we overload things too heavily, we can cause ambiguities:

```
function Empty return Sequence;
procedure Print (S : Sequence);
function Empty return Set;
procedure Print (S : Set);
…
Print (Empty); -- Illegal!
```

The call is ambiguous, and therefore illegal, because there are two possible meanings. One way to resolve the ambiguity is to use a qualified expression to say which type we mean:

```
Print (Sequence'(Empty));
```

Note that we're now using both bottom-up and top-down overload resolution: Sequence' determines which Empty is meant (top down) and which Print is meant (bottom up). You can qualify an expression, even if it is not ambiguous according to Ada rules — you might want to clarify the type because it might be ambiguous for human readers.

Of course, you could instead resolve the "Print (Empty)" example by modifying the source code so the names are unique, as in the earlier examples. That might well be the best solution, assuming you can modify the relevant sources. Too much overloading can be confusing. How much is "too much" is in part a matter of taste.

Ada really needs to have top-down overload resolution, in order to resolve literals. In some languages, you can tell the type of a literal by looking at it, for example appending "L" (letter el) means "the type of this literal is long int". That sort

of kludge won't work in Ada, because we have an open-ended set of integer types:

```
type Apple_Count is range 0..100;
procedure Peel (Count : Apple_Count);
…
Peel (20);
```

You can't tell by looking at the literal 20 what its type is. The type of formal parameter Count tells us that 20 is an Apple_Count, as opposed to some other type, such as Standard.Long_Integer. [Technically, the type of 20 is universal_integer, which is implicitly converted to Apple_Count -- it's really the result type of that implicit conversion that is at issue. But that's an obscure point -- you won't go _too_ far wrong if you think of the integer literal notation as being overloaded on all integer types.]

Programmers sometimes wonder why the compiler can't resolve something that seems obvious. For example:

```
type Apple_Count is range 0..100;
procedure Slice (Count : Apple_Count);
type Orange_Count is range 0..10_000;
procedure Slice (Count : Orange_Count);
…
Slice (Count => 10_000); -- Illegal!
```

This call is ambiguous, and therefore illegal. But why? Clearly the programmer must have meant the Orange_Count one, because 10_000 is out of range for Apple_Count. And all the relevant expressions happen to be static.

Well, a good rule of thumb in language design (for languages with overloading) is that the overload resolution rules should not be "too smart". We want this example to be illegal to avoid confusion on the part of programmers reading the code. As usual, a qualified expression fixes it:

```
Slice (Count => Orange_Count'(10_000));
```

Another example, similar to the literal, is the aggregate. Ada uses a simple rule: the type of an aggregate is determined top down (i.e., from the context in which the aggregate appears). Bottom-up information is not used; that is, the compiler does not look inside the aggregate in order to determine its type.

```
type Complex is
   record
      Re, Im : Float;
   end record;
procedure Grind (X : Complex);
procedure Grind (X : String);
…
Grind (X => (Re => 1.0, Im => 1.0)); -- Illegal!
```

There are two Grind procedures visible, so the type of the aggregate could be Complex or String, so it is ambiguous and therefore illegal. The compiler is not required to notice that there is only one type with components Re and Im, of some real type — in fact, the compiler is not _allowed_ to notice that, for overloading purposes.

We can qualify as usual:

```
Grind (X => Complex'(Re => 1.0, Im => 1.0));
```

Only after resolving that the type of the aggregate is Complex can the compiler look inside and make sure Re and Im make sense.

This not-too-smart rule for aggregates helps prevent confusion on the part of programmers reading the code. It also simplifies the compiler, and makes the overload resolution algorithm reasonably efficient.

How smart is "too smart" is in part a matter of taste. In fact, I would make the Ada rules a little bit less smart, if I were redesigning it from scratch. If we replaced the Grind on String procedure with:

```
procedure Grind (X : Integer);
```

then the above call would resolve, because the compiler _does_ use the fact that the aggregate must be some sort of aggregate-ish type, like a record or array. I would prefer the call to still be ambiguous in that case, but by and large, Ada gets the rules just about right, so something that is confusingly ambiguous to humans is usually ambiguous by the Ada rules.

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-sweden.org

Ada 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