ADA USER JOURNAL

Volume 29
Number 4
December 2008

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

Further details on our approach to these are given below.

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

News and Product Announcements

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

Reprinted Articles

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

Editorial

This December issue witnesses a change in the Editorial Team of the Ada User Journal. Santiago Urueña, who served as the News Editor for the past 5 years, is leaving the Journal, as he takes new responsibilities in his professional life. I would like to state my appreciation for Santiago's dedication and effort, and I wish him all the success in his new career.

Luckily, we were able to find another motivated young researcher to take over. I am thus pleased to welcome Marco Panunzio, a PhD student at the University of Padua, Italy, to the role of the Journal's News Editor. As Marco points out, he knows that this new duty will reduce his already non-existent free time, especially the little time he can find to play his piano, but he thinks that this is an interesting and worthy service to the Ada community.

Santiago and Marco jointly produced the news section in this issue, providing us with a smooth editor transition. I know the readers will not be disappointed.

As for the contents of the issue, we continue with the publication of material derived from the Ada-Europe 2008 conference. In this issue you can find three papers coming from the Industrial Track of the conference. The first paper, by Howard Ausden, of Lockheed Martin, USA, provides insights into the integration of Ada and C++ code in the ERAM air traffic control system. Afterwards, Jeff Cousins, of BAE Systems, UK describes the experience of porting a command and control system from Ada 95 to Ada 2005. And, finally, Jean-Pierre Rosen, of Adalog, France, presents the experience behind the specification of coding standards and programming rules for Ada programs. The technical part of the issue ends with an interesting set of Ada Gems, by Bob Duff and Ramón Fernandéz-Marina, related to Null (or not null) and Accessibility Checks.

I would like also to point out the rich set of events for which we provide information in the calendar and forthcoming events sections. Particularly, the latter provides information about five Ada related events in 2009; a plentiful year to look for.

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

News

***Santiago Urueña**[*] **and Marco Panunzio**[†]*

**Technical University of Madrid (UPM). Email: Santiago.Uruena@upm.es*
[†]University of Padua. Email: panunzio@math.unipd.it

Contents

Ada-related Organizations

XVII Award Ada-Spain

From: Ada-Spain
Subject: XVII award Ada-Spain for the best academic project related to the Ada programming language
URL: http://adaspain.unican.es/Premio_XVII.pdf

The goal of this award is to foster the adoption of the Ada programming language in University teaching programs as well as in professional training programs, both as study subject and as instrument to the realization of projects and investigation works.

The details of the award and the criteria for the eligibility of works follow:

1. It is hereby established an award granted with a price of 750 Euro for the winner and a price of 450 Euro for the runner-up.

2. The work can be realized by a group of persons, not exceeding the number of three persons.

3. The submitted projects shall be part of the academic work of their authors, who should be studying in an officially recognized teaching center (academic or for professional training) during the realization of the project. In particular, those works can be either projects assigned at the end of the study period or projects assigned during a course.

4. The deadline for the submission of the projects is February 20, 2009.

5. The submitted projects shall be sent via post courier to the following address:

Ada-Spain
Apartado Postal 50.403
28080 Madrid

[Translated from Spanish. For the complete announcement (in Spanish), please refer to http://adaspain.unican.es/Premio_XVII.pdf —mp]

Ada-related Events

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

Oct 30 — SIGAda Awards

From: John McCormick <mccormick@cs.uni.edu>
Date: Thu, 11 Sep 2008 06:37:20 −0700 (PDT)
Subject: 2nd Call for SIGAda Award Nominations
Newsgroups: comp.lang.ada

Dear Members of the Ada Community:

On Thursday, 30 October 2008, the 2008 SIGAda Awards will be presented in a special morning plenary session at the SIGAda 2008 conference in Portland, Oregon. (See http://www.acm.org/sigada/conf/sigada2008/ if you have somehow missed announcements of this year's annual SIGAda international conference.)

We welcome your nominations of deserving recipients.

The ACM SIGAda Awards recognize individuals and organizations who have made outstanding contributions to the Ada community and to SIGAda. The two categories of awards are:

(1) Outstanding Ada Community Contribution Award — For broad, lasting contributions to Ada technology & usage.

(2) ACM SIGAda Distinguished Service Award — For exceptional contributions to SIGAda activities & products.

Please consider who should be nominated this year. You may nominate a person for either or both awards, and as many people as you think worthy. One or more awards will be made in both categories.

Please visit http://www.acm.org/sigada/exec/awards/awards.html#Recipients and peruse the names of past winners. This may help you think about the measure of accomplishment that is appropriate. You may be aware of people who have made substantial contributions that have not yet been acknowledged. Nominate them. Consider what you believe to be the best developments in the Ada community or SIGAda in the last year; the last 5 years; since Ada's inception. Who was responsible? Nominate them.

Please note that anyone who has received either of the two awards remains eligible for the other. Perhaps there is an outstanding SIGAda volunteer who has won our Distinguished Service Award and who has also made important contributions to the advance of Ada technology, or vice versa. Nominate him or her!

The nomination form is available on the SIGAda website at http://www.acm.org/sigada/exec/awards/awards.html. (You need to visit this website to see past award winners' names, and also a picture of the statuette which is the award among other things, so you don't nominate someone who has already won an award in a category.) Submit your nomination as an e-mail or e-mail attachment to SIGAda-Award@ACM.ORG.

The ACM SIGAda Awards Committee, comprised of volunteers who have previously won an award, will determine this year's recipients from your nominations.

Call our attention to the people who are most deserving, by nominating them. And please nominate by SEPTEMBER 21!

Your participation in the nominations process will help maintain the prestige and honor of these awards.

Thank you,

John McCormick

Chair ACM SIGAda

Oct 26–30 — SIGAda 2008 Conference

From: Michael Feldman <mfeldman@seas.gwu.edu>
Date: Mon, 29 Sep 2008 19:47:30 -0500
Subject: SIGAda 2008 Conference Registration Deadlines Fast Approaching!
Newsgroups: comp.lang.ada

I just want to take a minute of your time to remind you of the ACM SIGAda 2008 International Conference, which will take place in Portland, Oregon, Oct. 26–30, 2008. The conference program is shaping up to be very interesting, and well worth the time and expense.

The deadline to reserve your hotel room at the special $99.00 conference rate is next Monday, Oct. 5, 2008!

See the conference details — program, registration, travel, hotel, etc. — at

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

Note that there's still time to apply for an educator grant which will cover your conference and tutorial registration.

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

FOSDEM 2009 Call for Interest

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Date: Wed, 8 Oct 2008 23:56:35 +0200 (CEST)
Subject: Ada at FOSDEM 2009 — Call for Interest
Organization: Ada-Belgium, c/o Dept. of Computer Science, K.U.Leuven
Keywords: Ada,open source,free software,technical presentations,FOSDEM
Newsgroups: comp.lang.ada,fr.comp.lang.ada

Call for Interest

A d a at F O S D E M 2 0 0 9

February 2009, Brussels, Belgium

FOSDEM [1], the Free and Open source Software Developers' European Meeting, is a free and non-commercial two-day event organized each February in Brussels, Belgium.

The goal is to provide Free Software and Open Source developers and communities a place to meet with other developers and projects, to be informed about the latest developments in the Free Software and Open Source world, to attend interesting talks and presentations by Free Software and Open Source project leaders and committers on various topics, and to promote the development and the benefits of Free Software and Open Source solutions.

In a Developers Room at FOSDEM 2006, Ada-Belgium [2] organized a very well attended full-day lecture program [3].

Each year the number of applications for DevRooms outnumbers the available space, presenting the organizers with a difficult selection [4]. For FOSDEM 2008, Ada-Belgium proposed another day of Ada presentations, but the organizers felt there was too little of an audience. We intend to propose again for FOSDEM 2009, and need to show that this would attract sufficient interest.

To increase our chances to be allocated a DevRoom, Ada-Belgium calls on you to:

- Speak loudly about the fact that you want to see Ada presentations at FOSDEM by sending email to info@fosdem.org (please CC ada-belgium-board@cs.kuleuven.be).
- Visit FOSDEM's brainstorm page [5] and propose Ada-related keynote speakers and topics (please let us know if you do).
- For bonus points, inform us at ada-belgium-board@cs.kuleuven.be about specific presentations you would like to hear in an Ada DevRoom.
- For more bonus points, subscribe to the Ada-FOSDEM mailing list [6] to discuss and help organize the details.
- For even more bonus points, be a speaker: the Ada-FOSDEM mailing list is the place to be!

We look forward to lots of feedback! Please act ASAP and definitely before November 15.

The FOSDEM Team of Ada-Belgium

PS: This Call for Interest is also available online [7], including versions in PDF format suitable for printing (152 KB) and in plain text format for further distribution (6 KB).

---

[1] http://www.fosdem.org

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

[3] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/06/060226-fosdem.html

[4] http://archive.fosdem.org/2008/call_for_devrooms

[5] http://www.fosdem.org/2009/brainstorm

[6] http://listserv.cc.kuleuven.be/archives/adafosdem.html

[7] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/09/0902xx-fosdem.html

Jun 8–12 — Ada-Europe 2009

From: dirk@heli.cs.kuleuven.be (Dirk Craeynest)
Date: Thu, 6 Nov 2008 23:51:16 +0100 (CET)
Subject: CfP 14th Conf. Reliable Software Technologies, Ada-Europe 2009
Newsgroups: comp.lang.ada,fr.comp.lang.ada, comp.lang.misc
Organization: Ada-Europe, c/o Dept. of Computer Science, K.U.Leuven
Keywords: Conference,tutorials,industry, reliability,Ada,LNCS,Brest,France

CALL FOR PAPERS

14th International Conference on

Reliable Software Technologies — Ada-Europe 2009

8 – 12 June 2009, Brest, France

Organized by Ada-Europe,

in cooperation with ACM SIGAda

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).

General Information

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

Schedule

01 December 2008: Submission of regular papers, tutorial and workshop proposals

12 January 2009: Submission of industrial presentation proposals

09 February 2009: Notification to all authors

09 March 2009: Camera-ready version of regular papers required

11 May 2009: Industrial presentations, tutorial and workshop material required

08–12 June 2009: Conference Topics

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

the completion of the Ada language standard revision process, contributions that present and discuss the potential of the revised language are particularly sought after.

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

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

Call for Regular Papers

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

Proceedings

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

Awards

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

Call for Industrial Presentations

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

Call for Tutorials

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

Call for Workshops

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

Call for Exhibitions

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

Grants for Students

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

Organizing Committee

Conference Chair

Frank Singhoff, UBO/LISyC, France

Frank.Singhoff@univ-brest.fr

Program Co-Chairs

Yvon Kermarrec, Télécom Bretagne, France

Yvon.Kermarrec@telecom-bretagne.eu

Fabrice Kordon, University Pierre & Marie Curie, France

Fabrice.Kordon@lip6.fr

Tutorial Chair

Jérôme Hugues, Télécom Paris-Tech, France

Jerome.Hugues@telecom-paristech.fr

Exhibition Chair

Pierre Dissaux, Ellidiss Technologies

Pierre.Dissaux@ellidiss.com

Publicity Chair

Dirk Craeynest, Aubay Belgium & K.U.Leuven, Belgium

Dirk.Craeynest@cs.kuleuven.be

Local Chairs

Alain Plantec and Mickael Kerboeuf, UBO/LISyC, France

Alain.Plantec@univ-brest.fr and Mickael.Kerboeuf@univ-brest.fr

Program Committee

Alejandro Alonso, Universidad Politécnica de Madrid, Spain

Leemon Baird, US Air Force Academy, USA

Johann Blieberger, Technische Universität Wien, Austria

Maarten Boasson, University of Amsterdam, The Netherlands

Bernd Burgstaller, Yonsei University, Korea

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

Michael González Harbour, Universidad de Cantabria, Spain

Javier Gutiérrez José, Universidad de Cantabria, Spain

Philippe Dhaussy, ENSIETA/LISyC, France

Andrew Hately, Eurocontrol CRDS, Hungary

Jérôme Hugues, Telecom Paris, France

Günter Hommel, Technischen Univesität Berlin, Germany

Hubert Keller, Institut für Angewandte Informatik, Germany

Yvon Kermarrec, Télécom Bretagne, France

Fabrice Kordon, Université Pierre & Marie Curie, France

Albert Llemosí, Universitat de les Illes Balears, Spain

Franco Mazzanti, ISTI-CNR Pisa, Italy

John McCormick, University of Northern Iowa, USA

Stephen Michell, Maurya Software, Canada

Javier Miranda, Universidad Las Palmas de Gran Canaria, Spain

Scott Moody, Boeing, USA

Daniel Moldt, University of Hamburg, Germany

Laurent Pautet, Telecom Paris, France

Laure Petrucci, LIPN, Université Paris 13, 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

Lionel Seinturier, Université de Lille, France

Frank Singhoff, UBO/LISyC, France

Oleg Sokolsky, University of Pennsylvania, USA

Ricky Sward, MITRE, USA

Tullio Vardanega, Università di Padova, Italy

Francois Vernadat, LAAS-CNRS, Université de Toulouse, Insa

Andy Wellings, University of York, UK

Jürgen Winkler, Friedrich-Schiller-Universität, Germany

Luigi Zaffalon, University of Applied Sciences, W. Switzerland

Industrial Committee

Guillem Bernat, Rapita Systems, UK

Agusti Canals, CS, France

Roderick Chapman, Praxis HIS, UK

Colin Coates, Telelogic, UK

Dirk Craeynest, Aubay Belgium & K.U.Leuven, Belgium

Dirk Dickmanns, EADS, Germany

Tony Elliston, Ellidiss Software, UK

Franco Gasperoni, AdaCore, France

Hubert Keller, Forschungszentrum Karlsruhe GmbH, Germany

Bruce Lewis, US Army, USA

Ahlan Marriott, White-Elephant GmbH, Switzerland

Rei Stråhle, Saab Systems, Sweden

CfP in PDF format

http://www.cs.kuleuven.be/~dirk/ada-belgium/events/09/090608-aec-cfp.pdf

Feb 7–8 — Ada Developer Room at FOSDEM 2009

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Date: Mon, 1 Dec 2008 22:15:53 +0100 (CET)
Subject: Ada Developer Room at FOSDEM 2009
Organization: Ada-Belgium, c/o Dept. of Computer Science, K.U.Leuven
Keywords: Ada,open source,free software,technical presentations,FOSDEM
Newsgroups: comp.lang.ada

Preliminary Announcement

Ada Developer Room at FOSDEM 2009

7 – 8 February 2009, Brussels, Belgium

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

FOSDEM, the Free and Open source Software Developers' European Meeting, is a free and non-commercial two-day event organized each February in Brussels, Belgium.

We are very pleased to announce that the organizers of FOSDEM 2009 have accepted our proposal for an Ada Developer Room at the next event, i.e. on Sat 7 and Sun 8 February 2009.

The full list of presentations and speakers is available on the Ada at FOSDEM 2009 web-page. More details, such as the concrete schedule, will follow later.

We hope to see many of you there!

Valentine, Ludovic, Dirk

The FOSDEM Team of Ada-Belgium

Avionics USA 2009

From: AdaCore Press Center
Date: Wednesday November 26, 2008
Subject: Avionics USA 2009
RSS: http://www.adacore.com/2008/11/26/avionics-usa-2009/

SSTC

From: AdaCore Press Center
Date: Wednesday November 26, 2008
Subject: Systems & Software Technology Conference (SSTC)
RSS: http://www.adacore.com/2008/11/26/systems-software-technology-conference-sstc/

ESC Silicon Valley

From: AdaCore Press Center
Date: Wednesday November 26, 2008
Subject: ESC Silicon Valley
RSS: http://www.adacore.com/2008/11/26/esc-silicon-valley/

AdaCore — Lean Event

From: AdaCore Press Center
Date: Wednesday November 26, 2008
Subject: Lean Event
RSS: http://www.adacore.com/2008/11/26/lean-event/

If you are interested in learning more about this event, please contact events@adacore.com

AdaCore is main organizer and sponsor of this event. Talks will be given by Jim Sutton (author Lean Software Strategies), Emmanuel Chenu (Thales Avionics), Andy Vickers (Praxis High Integrity Systems), and Cyrille Comar (AdaCore).

Ada Conference UK

From: AdaCore Press Center
Date: Wednesday November 26, 2008
Subject: Ada Conference, UK.
RSS: http://www.adacore.com/2008/11/26/ada-conference-uk/

AdaCore is event co-organizer and main sponsor. Franco Gasperoni will be giving a talk entitled “Project Coverage and Open-DO”.

Avionics 09 Europe

From: AdaCore Press Center
Date: Wednesday November 26, 2008
Subject: Avionics 09 (Europe)
RSS: http://www.adacore.com/2008/11/26/avionics-09-europe/

AdaCore is the major sponsor of the Military Avionics track and will be co-hosting the workshop “Worst-Case and Structural Coverage Analysis — Tools and Technologies to get DO-178B” with Wind River.

Ada Semantic Interface Specification (ASIS)

Gela ASIS

From: Maxim Reznik <reznikmm@gmail.com>
Date: Fri, 19 Sep 2008 09:13:24 −0700 (PDT)
Subject: Announce: Gela ASIS 0.2
Newsgroups: comp.lang.ada

I would like to announce Gela ASIS 0.2 release.

http://www.ten15.org/wiki/gela_asis/

Gela ASIS is platform/compiler independent implementation of Ada Semantic Interface Specification (ASIS). Gela ASIS implements core ASIS Version 2.0 and most of ASIS Issues (SI99), so it's capable to process Ada 2005 code.

Recent changes are:

- Makefile to easy build and install Gela ASIS
- implementation of Ada 2005 issues
- implementation of private operations
- implementation of Asis.Compilation_Units.Relations

Ada and Education

Praxis — SPARK Training March 2009

From: Praxis HIS — SPARKAda
Subject: SPARK Training
Date: March, 2009
URL: http://www.praxis-his.com/sparkada/training.asp

SPARK Training — Overview

We run four courses in SPARK, details of which are below. The schedule for public courses is shown below. Exclusive courses for clients, either at our offices or on-site, are also available — please contact us for details.

Course 1: Software Engineering with SPARK

A 4-day course for managers, regulators and engineers, which presents the principles of the development of high integrity software, and the related certification requirements. It then explains the rationale of SPARK, outlines the language and the principles of static code analysis, and presents the role of the SPARK Examiner in systematic program development. The course also covers fundamental SPARK design issues, such as appropriate use of packages such as abstract machines and data types, as well as the use of SPARK refinement, system interfaces, library mechanisms, etc. Some of the more advanced facilities of the SPARK Examiner, for run-time error checking for example, are presented.

Course 2: Black-Belt SPARK

A course for engineers who have already attended the “Software Engineering with SPARK” course or are experienced SPARK users. This course covers the advanced use of SPARK, particularly in the context of proof of exception freedom and code correctness. Attendees are taught to understand the relationship between SPARK source code and the verification conditions generated for proof, leading to an understanding of the impact of good SPARK design principles on code verification. Advanced facilities of the SPARK Examiner are presented, and tuition in planning, conducting and managing the verification activities is supplemented by the use of the SPARK proof tools, particularly the Simplifier. The course has a strongly practical flavour, interweaving guidance and lecture material with topical tutorial sessions which reinforce the lecture material via relevant examples. Each tutorial session commences with a step-by-step example which provides detailed guidance, followed by additional exercises which can be tried in the tutorial sessions or used after the course to gain additional practical experience.

Course 3: High-Integrity Concurrent Software Design with RavenSPARK

The Ada 95 Ravenscar profile defines a subset of the Ada 95 tasking facilities that are appropriate for the construction of high-integrity software. This one-day course introduces the Ravenscar profile and how it has been included in the core SPARK language. The course will cover the additional annotations in SPARK that are used to describe packages that contain tasks and protected objects and the additional analyses implemented by the Examiner to eliminate the potential for defects in Ravenscar programs.

Delegates for this course should have already attended the introductory “Software Engineering with SPARK” course, or should be experienced SPARK users. This course may be taken as a one-day stand-alone module, or may directly follow a “Software Engineering with SPARK” course.

Course 4: UML to SPARK

This course covers the rationale for integrating SPARK with UML, and the generation of SPARK from UML. The majority of the course consists of a practical session, where delegates will produce SPARK from a partially completed UML model.

Delegates for this course should have already attended the introductory “Software Engineering with SPARK” course, or should be experienced SPARK users. No knowledge of UML or experience of using UML tools is assumed. This course may be taken as a one-day stand-alone module, or may directly follow a “Software Engineering with SPARK” course.

Public Course Dates for 2009 — UK

Course 1 — “Software Engineering with SPARK”

2nd to 5th March 2009, Bath, UK.

Course 2 — “Black-Belt SPARK”

17th to 19th March 2009, Bath, UK.

Course 3 — “High-Integrity Concurrent Software Design with RavenSPARK”

TBD — come back soon for future course dates.

Course 4 — “UML to SPARK”

TBD — come back soon for future course dates.

Courses in the USA

Praxis High Integrity Systems can run training courses at a customer's facilities as required. Training in the USA is also available from our partner company Pyrrhus Software.

Courses in Australia

Please contact us for details of training in Australia.

Enquiries and Reservations

For enquiries and reservations for the course, please contact us.

[See also the same topic in AUJ 28-3 (Sep 2007) —mp]

AdaCore — Talk on performance and genericity

From: AdaCore Press Center
Date: Friday October 24, 2008
Subject: Performance and Genericity Seminar
RSS: http://www.adacore.com/2008/10/24/perfrmance-and-genericity-seminar/

Thomas Quinot will be giving a talk “Efficient representation of complex data in schizophrenic middleware”.

AdaCore — GPS 4.3 Webinar available for download

From: AdaCore Developer Center
Date: Thursday November 27, 2008
Subject: GPS 4.3 InSight webinar now available for viewing
RSS: http://www.adacore.com/2008/11/27/gps-43-insight-webinar-now-available-for-viewing/

The latest GNAT Pro InSight webinar featuring GPS 4.3 is now available for viewing. The webinar included a presentation and demo of some of the following features:

- Easy configuration of dual compilation.
- Enhanced support for gcov (code coverage), gnatcheck (coding standard checker) and compiler switches.
- A redesigned and fully customizable builder module.
- New plug-ins.
- Improved code completion.
- Improved documentation generator.

To view this event, please click here or visit www.adacore.com/home/gnatpro/webinars

Ada-related Resources

AdaCore — Ada XML Library added to Live Docs

From: AdaCore Developer Center
Date: Tuesday September 30, 2008
Subject: XML Ada Library documentation added to Live Docs
RSS: http://www.adacore.com/2008/09/30/2380/

The XML Ada Library documentation has been added to Live Docs. The object-oriented XML/Ada library allows development of applications for parsing and processing XML streams, with a SAX implementation that that allows conversion of such streams into application-specific data representations. Live Docs provides an up to the minute snapshot of GNAT Pro technology. As new features and improvements are made to GNAT Pro these changes are immediately added to our product documentation and presented here in Live Docs.

More info on XML/Ada, please visit:

http://www.adacore.com/home/gnatpro/add-on_technologies/xml_ada/

To view the Live Docs page, please visit:

http://www.adacore.com/category/developers-center/

reference-library/documentation

[http://www.adacore.com/wp-content/files/auto_update/xmlada-docs/xml.html —su]

AdaCore — GPRbuild documentation added to Live Docs

From: AdaCore Developer Center
Date: Tuesday October 7, 2008
Subject: GPRbuild documentation added to Live Docs
RSS: http://www.adacore.com/2008/10/07/gprbuild-documentation-added-to-live-docs/

The GPRbuild documentation has been added to Live Docs. GPRbuild is an advanced software tool designed to designed to automate the construction of multi-language systems using GNAT Project files. Live Docs provides an up to the minute snapshot of GNAT Pro technology. As new features and improvements are made to GNAT Pro these changes are immediately added to our product documentation and presented here in Live Docs.

[http://www.adacore.com/wp-content/files/auto_update/gprbuild-docs/html/gprbuild_ug.html —su]

AdaCore — GNATcoll documentation added to Live Docs

From: AdaCore Developer Center
Date: Monday October 20, 2008
Subject: GNATcoll documentation added to Live Docs
RSS: http://www.adacore.com/2008/10/20/gnatcoll-decumentation-added-to-live-docs/

The GNAT Components Collection (GNATcoll) documentation has been added to Live Docs. The GNAT Components Collection is a suite of reusable software components and utilities originating from the AdaCore infrastructure. Live Docs provides an up to the minute snapshot of GNAT Pro technology. As new features and improvements are made to GNAT Pro these changes are immediately added to our product documentation and presented here in Live Docs.

Ada-related Tools

GNAT Pro for .NET

From: Rob Veenker <veenker@xs4all.nl>
Date: Sun, 23 Nov 2008 17:36:42 +0100
Subject: Re: GNAT Pro for .NET
Newsgroups: comp.lang.ada

> Wonder if anyone has any experience with Ada .NET? How well it is integrated and what are the major issues you had with it?

You may want to check out the MGNAT project at http://sourceforge.net/projects/asharp

The GNAT Pro for .Net compiler used this project as a starting point.

The compiler is kept in sync with the latest GNAT Pro compiler technology but there are limitations when it comes to code generation (due to .Net runtime). Also note that not all annexes are implemented.

You need to be aware of the way the Ada symbols (variables, procedures, functions etc.) are expressed in MSIL if you want to create Ada assemblies that will be used by other .Net languages. Using a .Net reflection tool or the object browser can be of great help.

Integration in the MS Visual Studio is great for debugging a mixed language application, but for my Ada programming I still use GPS.

From: Rob Veenker <veenker@xs4all.nl>
Date: Tue, 25 Nov 2008 00:07:35 +0100
Subject: Re: GNAT Pro for .NET
Newsgroups: comp.lang.ada

> Thanks, Rob. I've tried that, but there are some strange things in the way Ada objects behave and with lack of documentation it's hard to figure out. In particular:
>
> - Is it possible to mix native code from Ada with MSIL (like we do C++/C# mix)?

I have not tested the integration in passing Ada objects but there is a way of extending MSIL classes using Ada. I believe there is even a tutorial on this.

To be able to call 'native' Ada code I still rely on the MS COM+ interface using GNATCOM and wrap this native COM object using interop services.

But you can also use pragma import / export !

> - I was unable to get Initialize/Finalize being called for Ada.Finalization.Controlled objects (given that destruction of objects in .NET are non-deterministic, but still).
> - Compiler won't let me implement IDisposable interface, am I looking at right place (MSSyst.IDisposable)?

Have a look at the extend_dotnet_class tutorial. Here the MSSyst.IDisposible is indeed used as well. But the object is created outside of Ada. Which version of Ada for .Net are you using? There should be something on this subject in the documentation as well.

From: Brad Moore <brad.moore@shaw.ca>
Date: Wed, 26 Nov 2008 00:05:51 −0700
Subject: Re: GNAT Pro for .NET
Newsgroups: comp.lang.ada

I've been involved in a port of a number-crunching system originally written in

Ada 83 from a different compiler vendor running on Unix, to GNAT on Windows in Ada 95, and then most recently to .NET in Ada 2005 using GNAT Pro for .NET. I say port, but for the most part it was simply a matter of setting the compiler switches for Ada 95, and Ada 2005, and recompiling, though we have been introducing new language features during maintenance of the system.

The original system features several tasks with rendezvous, and heavy use of the Ada math libraries. There is no GUI involved.

My experiences from these ports are;

Port 1) Ada 83 SCO Unix (non-GNAT vendor) to Win32 GNAT in Ada 95

- Issues relating to compiler vendor switch:

  None come to mind

- Issues relating to language switch, Ada 83 — Ada 95

  1) We had a generic protected queue, which had used the Ada 95 reserved word “protected” in its variable names. We had to change the name of the variables to something else. (A very trivial change)

  2) The generic would accept an unconstrained type as a formal parameter. In Ada 95, we had to add the box notation to the formal parameter.

  eg. Instead of;

```
generic
    type Element_Type is private;
package P is
```

  we had to write;

```
generic
    type Element_Type (<>) is
      private;
package P is
    …
```

  Another very trivial change.

  3) We had used passive tasks in a couple of places. In Ada 95, we replaced these library units with protected types. This was a bit more work, but the end result was satisfying and a better result.

- Issues relating to operating system switch: Unix to Windows

  This was probably the biggest impact, though we had isolated OS routines, so changes were quite localized. The biggest challenge was finding a replacement for Unix's memory mapped array feature, which Windows didn't seem to have matching support for.

- Issues relating to change from services being invoked by CORBA to interface accessed via a DLL.

  The use of the Interfaces.C library routines was very helpful. To invoke the DLL from .NET, a .NET wrapper was needed to complete the interface.

Port 2: Ada 95 Windows to Ada 2005 .NET

- Issues relating to the language switch. (Ada 95 to Ada 2005)

  None: Or at least I don't recall there being any.

  We have now incorporated new language features including interfaces, Ada.Directories, containers, object prefix notation, to name a few. These mostly come as enhancements needed for maintenance, and are unrelated to the actual port.

- Issues going from win32 to .NET

  For the most part we didn't need to make any significant code changes. There are a few libraries that weren't implemented in .NET Ada. The ones that come to mind are Ada.Direct_IO and Ada.Directories I believe, though there are other libraries that come with GNAT that provided suitable workarounds. There may have been some other minor issues, but we found workarounds for everything. The application runs well in .NET. We no longer needed the .NET wrapper around our DLL, which simplified things. We had to get rid of our Interfaces.C pragmas and replace with similar vendor specific pragmas for interfacing to .NET. I believe the full list of issues/differences between .NET GNAT Pro and Windows GNAT Pro can be obtained from AdaCore.

JEWL on XP

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Mon, 10 Nov 2008 01:50:57 GMT
Subject: Re: JEWL on XP?
Newsgroups: comp.lang.ada

> From a quick scan of the documentation, JEWL would appear to provide all I need. However certain parts of that same documentation lead to the question, will JEWL run on XP? And if not, is there anything else that might serve?

JEWL runs fine on XP.

Depending on your compiler, you may need to convert the single “pragma Linker_Options” into multiple, one for each item in the single one.

From: John McCormick <mccormick@cs.uni.edu>
Date: Mon, 10 Nov 2008 06:32:52 −0800 (PST)
Subject: Re: JEWL on XP?
Newsgroups: comp.lang.ada

Here are the details of the changes I give to my students to run JEWL on XP

In file

jewl-win32_interface.adb

replace

```
pragma Linker_Options ("-luser32
  -lgdi32 -lcomdlg32 -lwinmm");
```

with

```
pragma Linker_Options ("-luser32");
pragma Linker_Options ("-lgdi32");
pragma Linker_Options ("-lcomdlg32");
pragma Linker_Options ("-lwinmm");
```

Build the JEWL library as described in the JEWL documentation.

From: “Randy Brukardt” <randy@rrsoftware.com>
Date: Mon, 10 Nov 2008 19:14:27 −0600
Subject: Re: pragma Linker_Options (was: JEWL on XP?)
Newsgroups: comp.lang.ada

>> Depending on your compiler, you may need to convert the single “pragma Linker_Options” into multiple, one for each item in the single one.

So, pragma Linker_Options can be troublesome in case one cannot change the Ada source files. (Because whenever a compilation system interprets the string_expression of the pragma and the linker does not understand the string, compilation fails.) Wouldn't it be better, in general, to avoid system dependent pragma Linker_options in source files?

Well, maybe, but then you potentially lose version control on the options. And you might not have control over the building environment, but still need the options given.

For instance, in Claw, we have a package with separate bodies for each compiler. Some of those bodies contain one or more Linker_Option pragmas. If they're not given, Claw won't work (usually with a completely mysterious error message). We don't provide build scripts with Claw (the various compilers have fine mechanisms for that, we don't need to duplicate that). So pragma Linker_Options is the only way to ensure that the correct options are used.

I'd expect issues like these to arise for any “binding” packages that are distributed separately (assuming that you need facilities beyond those provided with the core OS). Similar reasons arise as to why you might want to use configuration pragmas rather than compiler options to suppress checks and the like.

QtAda bindings

From: Vadim Godunko <vgodunko@gmail.com>
Date: Tue, 9 Sep 2008 11:24:13 −0700 (PDT)
Subject: Announce: QtAda 2.0.0
Newsgroups: comp.lang.ada

We are pleased to announce QtAda 2.0.0 release.

Multi-platform source code package and Microsoft Windows binary package of the QtAda 2.0.0 can be downloaded from:

http://www.qtada.com/

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

New in QtAda 2.0.0:

- improved integration with Qt meta system allows to simplify user's code.
- non-GNAT users and GCC/GNAT users who don't have ASIS implementation may use build-in ASIS implementation from the Gela project.
- exception propagation and handling was improved on Microsoft Windows platform.
- and many other changes…

Ada-autoit

From: Per Sandberg <per.sandberg@bredband.net>
Date: Tue, 21 Oct 2008 23:49:15 +0100
Subject: [ANN] ada-autoit updated to work with AUtoIt v3.2.12
Newsgroups: comp.lang.ada

ada-autoit is a medium thick Ada-binding to the scripting tool Auto It.

Note this tool is very win32 specific.

For more info see:

http://www.autoitscript.com/autoit3/index.shtml

and:

http://sourceforge.net/projects/ada-autoit/

From: Steve <steved94@comcast.net>
Date: Wed, 22 Oct 2008 18:53:01 −0700
Subject: Re: [ANN] ada-autoit updated to work with AUtoIt v3.2.12
Newsgroups: comp.lang.ada

I recently tried using autoit and found it interesting. Unfortunately it wasn't able to do everything I wanted to do with out a lot of extra work, or at least learning a new programming language (the special dialect of basic used by autoit). I did a little research to see if there was an easier way and found the Microsoft UI Automation Library.

http://msdn.microsoft.com/en-us/magazine/cc163288.aspx

For my app I wound up using the C# and the .NET framework which contains an automation framework (system.windows.automation). If I were developing the same app in Ada I would either use A# or create a binding to the Microsoft UI Automation Library using GnatCOM.

From: Per Sandberg <per.sandberg@bredband.net>
Date: Thu, 23 Oct 2008 08:04:18 +0100
Subject: Re: [ANN] ada-autoit updated to work with AUtoIt v3.2.12
Newsgroups: comp.lang.ada

A short reply.

*) Yes Autoit is a quite simple and there are a lot of things it don't do.

*) The drive behind the binding was simplicity.

*) A# is out of the question for several reasons although it might work very well but I got a lot of legacy stuff and just want to press a few buttons.

*) What i found using GnatCOM directly is that I end up with a lot of code in the application layer and usually have to write some kind of wrapper on the generated code anyway (to leave the COM-World).

So I usually find it simpler to translate from the C header files directly to low level Ada by hand (basically what i will get from GnatCOM) and then do the higher level Ada mapping.

Of course all this depends on the size and complexity of the interface.

From: Mikhail Terekhov <terekhov@emc.com>
Date: Fri, 24 Oct 2008 22:56:01 −0400
Subject: Re: [ANN] ada-autoit updated to work with AUtoIt v3.2.12
Newsgroups: comp.lang.ada

You may find this link interesting as well http://pywinauto.openqa.org

CairoAda 1.8

From: Damien Carbonne <damien.carbonne@free.fr>
Date: Sun, 23 Nov 2008 19:07:13 +0100
Subject: ANN: CairoAda-1.8
Newsgroups: comp.lang.ada

CairoAda is now available here:
http://sourceforge.net/projects/cairoada/

This version is compliant with Cairo-1.8.x

GNADE and ODBC interfacing

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 3 Oct 2008 06:57:24 −0700 (PDT)
Subject: Re: GNADE/ODBC connection problem
Newsgroups: comp.lang.ada

> I'm trying to use GNADE to connect to a MySQL database but i keep getting the same error:
>
> raised GNU.DB.SQLCLI.DRIVER_ERROR : [Proc=3DSQLConnect][Server=3D][State=3DIM002][unixODBC][Driver Manager]Data source name not found, and no default driver specified

Have you configured your odbc.ini file? GNADE looks there for the data source name you specified in your Ada program and for the driver it should load to access the database. The configuration of odbc.ini is outside the scope of GNADE as this is really part of UnixODBC (which, of course, I assume you have installed).

From: Maciej Sobczak <maciej@msobczak.com>
Date: Sat, 4 Oct 2008 13:54:28 −0700 (PDT)
Subject: Re: GNADE/ODBC connection problem
Newsgroups: comp.lang.ada

Depending on your needs, the following library might be enough:

http://www.inspirel.com/soci-ada/

These are *complete* program examples:

http://www.inspirel.com/soci-ada/doc/idioms.html

The SOCI library uses native interface for MySQL, so there is nothing to configure.

OpenToken

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 12 Sep 2008 20:40:30 +0200
Subject: Re: OpenToken
Newsgroups: comp.lang.ada

Ted Dennison, the author of OpenToken, reappeared yesterday on the AWS mailing list and I took the opportunity to send him an email which he allowed me to reproduce here for the benefit of all interested. The conversation is below.

Ludovic Brenta wrote:

> Are you the author of OpenToken? There has been a discussion on comp.lang.ada starting at [1] about adopting it for future maintenance. As it turns out, Stephe Leake is willing to take it over. I'd like to know if you were aware of this and of the fact that, over the years since you released 3.0b, several people sent you patches which you never acknowledged. Do you approve if someone else takes over?
>
> [1] http://groups.google.com/group/comp.lang.ada/msg/ef40447ce799fba1

Ted Dennison replied:

> I'm unaware of the talk. I am quite aware that I've been sent a few patches that I never had time to incorporate, and

> that it's not being actively developed. The birth of my second child pretty much killed all the free time I had to do such things. I now have a third, so the free time situation is even worse.
>
> I certainly approve, heartily, of anyone taking over development of it. The whole point of licensing it the way I did was so that such things could happen.
>
> One suggestion I would make to people would be to use a public source code repository. Among other things, that would make it much easier to distribute the burden of testing and incorporating patches. If I were starting such a project today, I'd definitely use Git for revision control. It works fine in Windows now, and Git makes forking around developers who get busy/lazy and drop out (such as myself) nearly trivial. I believe Savannah supports it, as do a few other lesser-known public hosting sites: http://git.or.cz/gitwiki/GitHosting.

and then in a second email:

> Ludovic Brenta wrote:
>
>> Thanks a lot. Can I forward your reply to comp.lang.ada for the benefit of all?
>
> Certainly.
>
>> The agreement is to use Ada-France's monotone server. Monotone is also distributed, like git, but simpler to use, and it is written with the Ada attitude whereas C is written with the C attitude :)
>
> Interesting. I'll have to look into that.

I just imported OpenToken versions 2.0 and 3.0b into the Ada-France database under the branch name “org.opentoken”. You can browse it at http://www.ada-france.org:8081/branch/changes/org.opentoken

[…]

Status of AdaBrowse

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 24 Nov 2008 00:47:10 −0800 (PST)
Subject: Re: Status of AdaBrowse
Newsgroups: comp.lang.ada

> Just checking on the status of AdaBrowse. The latest release (4.0.3) appears to be from 2005. Has development of this stopped?

I think it has but anyone can take it over; it's at SourceForge.

Ada on Mac

From: Michael Feldman <mfeldman@gwu.edu>
Date: Wed, 10 Sep 2008 18:28:53 −0700
Subject: Re: Ada on Mac
Newsgroups: comp.lang.ada

> Finally jumped ship and got a Mac…what's the best, free (as in beer) Ada environment for Mac OS X?
>
> […]

See www.macada.org. Download the Leopard installer for GNAT and you're up and running. Make sure you've installed Apple's Developer Tools first, especially the XCode tools. They're in the optional-installs folder on your system disk.

GNAT/Mac comes with a plug-in for XCode (which is Apple's standard developer IDE). It's pretty much like all the other IDEs you've seen.

From: Jerry <lanceboyle@qwest.net>
Date: Thu, 11 Sep 2008 01:49:19 −0700 (PDT)
Subject: Re: Ada on Mac
Newsgroups: comp.lang.ada

[…]

I agree about Xcode.

www.macada.org is quirky so make sure you sign up for the mailing list if you have any problems.

There are a couple of free editors with IDE features of note that are Ada-friendly. Check out Smultron and TextWrangler. Not free ($49) but highly recommended is TextMate.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Thu, 11 Sep 2008 16:12:46 +0200
Subject: Re: Ada on Mac
Newsgroups: comp.lang.ada

> Doesn't Emacs work on Mac ?

Yes, and a recent edition of Emacs runs “graphically” on Mac OS X, not just in a terminal window.

Eclipse runs well, too.

From: John B. Matthews <jmatthews@wright.edu>
Date: Fri, 12 Sep 2008 22:08:07 −0400
Subject: Re: Ada on Mac
Newsgroups: comp.lang.ada

[…]

If you like TextWrangler, here's a keyword module for syntax coloring:

http://home.woh.rr.com/jbmatthews/misc/bbedit.html

From: “Martin Krischik” <krischik@users.sourceforge.net>
Date: Mon, 15 Sep 2008 21:34:29 +0200
Subject: Re: Ada on Mac
Newsgroups: comp.lang.ada

> Finally jumped ship and got a Mac…what's the best, free (as in beer) Ada environment for Mac OS X?

Funny that you ask, I just created one:

http://sourceforge.net/project/showfiles.php?group_id=12974&package_id=291480

needs http://www.macports.org/ to be installed first.

But then there is also MacAda:

http://www.macada.org/

Which works with XCode.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Mon, 29 Sep 2008 03:56:01 −0700 (PDT)
Subject: Re: Ada on Mac
Newsgroups: comp.lang.ada

The Wikibook page at http://en.wikibooks.org/wiki/Ada_Programming/Installing lists both macada and the GNU Ada project in separate sections (resp. 4.5.9 and 4.3 in the table of contents) but it doesn't list Max OS as supported in the GNU Ada project, nor does it explain the differences between the two. Could someone in the know please add the appropriate information for the benefit of all?

In particular I am curious how a hypothetical programmer wanting to program in Ada on their Macintosh would choose between the two. Are there differences in licensing? Integration with the rest of the system? Libraries included?

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Tue, 30 Sep 2008 07:58:06 +0200
Subject: Re: Ada on Mac
Newsgroups: comp.lang.ada

> The Wikibook page at http://en.wikibooks.org/wiki/Ada_Programming/Installing lists both macada and the GNU Ada project in separate sections (resp. 4.5.9 and 4.3 in the table of contents) but it doesn't list Max OS as supported in the GNU Ada project, nor does it explain the differences between the two. Could someone in the know please add the appropriate information for the benefit of all?

Well my hope was to upstream to MacPorts in which case a MacPorts section would be needed — and then listing differences would be interesting.

Currently GNU Ada is just staging area for a MacPosts version.

However, my upstream patches where not received well:

https://trac.macports.org/ticket/16549

Plan B would be a fork either as a part of MacPorts or inside GNU Ada Project.

> In particular I am curious how a hypothetical programmer wanting to program in Ada on their Macintosh would choose between the two. Are there differences in licensing? Integration with the rest of the system? Libraries included?

Once here truly is a MacPorts version, the differences would be explained in the line of integration: MacAda aims to integrate GNAT with the existing Mac OS X development environment (read XCode) while MacPorts is about integrating Mac OS X with the rest of the Unix world (read “configure make install”).

MacPorts

From: "Martin Krischik" <krischik@users.sourceforge.net>
Date: Mon, 15 Sep 2008 21:31:46 +0200
Subject: [Announcement] MacPorts / Mac OS X Release for GCC 4.3.2
Newsgroups: comp.lang.ada

I created a first test version of an MacPorts [1] / Mac OS X version of GNAT. Please donwload here to test:

http://sourceforge.net/project/showfiles.php?group_id=12974&package_id=291480

The aim of MacPorts is portability between Mac OS X and the rest of the Unix word. Here it differentiates from MacAda — which aims for perfect OS X integration.

From: Jerry <lanceboyle@qwest.net>
Date: Mon, 15 Sep 2008 13:15:47 −0700 (PDT)
Subject: Re: MacPorts / Mac OS X Release for GCC 4.3.2
Newsgroups: comp.lang.ada

Will this compiler work as (1) Ada 2005 on (2) PPC running (3) OS X 10.5?

I committed to supporting the PLplot bindings on both Ada 95 and Ada 2005, and I have about the last PPC Powerbook (laptop) sold by Apple so I'm not eager to buy a new computer just to get a compiler that runs on Intel. However, there has been no Ada 2005 compiler that runs under OS X 10.5 on PPC architecture. (The MacAda group has not been helpful in this regard.) As a result, I haven't been able to upgrade to 10.5.

Does your build support 10.5 on PPC, or do you know of other builds that will?

From: Simon Wright <simon.j.wright@mac.com>
Date: Mon, 15 Sep 2008 23:45:46 +0100
Subject: Re: MacPorts / Mac OS X Release for GCC 4.3.2
Newsgroups: comp.lang.ada

> However, there has been no Ada 2005 compiler that runs under OS X 10.5 on PPC architecture. (The MacAda group has not been helpful in this regard.) As a result, I haven't been able to upgrade to 10.5.

Do you think there's any chance that the GNAT/GPL 2007 compiler I built for Tiger/PPC would run on Leopard? It uses static libraries, maybe there's a chance… I moved to Intel Mac after that, so never upgraded the Powerbook to Leopard.

http://gnuada.sourceforge.net/pmwiki.php/Install/MacOS-GPL

From: "Martin Krischik" <krischik@users.sourceforge.net>
Date: Tue, 16 Sep 2008 07:52:24 +0200
Subject: Re: MacPorts / Mac OS X Release for GCC 4.3.2
Newsgroups: comp.lang.ada

MacPorts is a source based distribution — the binary package is just a convenience. So yes, it might work.

[…]

You need an older working compiler to create the up to date version. So a little tweaking will be needed.

Ada-related Products

AdaCore — GNATbench

From: AdaCore Press Center
Date: October 2, 2008
Subject: GNATbench 2.2.0 + 2.1.1
RSS: http://www.adacore.com/2008/10/02/gnatbench-220-211/

Announcing the availability of new versions of GNATbench

AdaCore is pleased to announce GNATbench 2.2.0 for Eclipse and GNATbench 2.1.1 for Wind River's Workbench.

The latest version of the GNAT Pro Eclipse-based plug-in is available for the following hosts:

x86-windows
sparc-solaris
x86-linux
x86_64-linux

Version 2.2.0 of GNATbench for Eclipse now supports Eclipse 3.4, with version 5.0.x of the C/C++ Development Tools (CDT), on the Linux (x86 and x86-64), Solaris (SPARC), and Windows platforms.

New features include:

- New GNAT Project explorer
- Separation between public and private API
- Better handling of scenario variables
- Better handling of import wizard errors

Version 2.1.1 of GNATbench for Workbench now supports Workbench 3.0 and is available on the following platforms:

x86-windows
sparc-solaris
x86-linux

and introduces the following features and enhancements:

- Improved project management and presentation
- An enhanced language-sensitive editor
- Additional wizards
- An improved builder
- Source code navigation enhancements

AdaCore — GNAT Pro for Nucleus OS

From: AdaCore Press Center
Date: Monday October 27, 2008
Subject: AdaCore Announces GNAT Pro for Nucleus OS
RSS: http://www.adacore.com/2008/10/27/nucleus-os/

Ada development support for Nucleus brings reliability to high-volume, embedded computing

BOSTON, October 28, 2008 — Embedded Systems Conference — AdaCore, provider of the highest quality Ada tools and support, today announced the availability of GNAT Pro for Nucleus® OS, the embedded operating system from Mentor Graphics. Nucleus OS offers a highly configurable kernel and utility extensions, making it a popular choice for small, high-volume embedded computing applications where low cost and high reliability are critical. With GNAT Pro for Nucleus OS, embedded system developers can increase their productivity through the reliability of the Ada language, the frontline support from AdaCore, and the capabilities of the GNAT Pro toolchain.

Nucleus OS provides an efficient, fast, deterministic and highly configurable operating system environment. Since the Nucleus OS is directly linked with the final application, the memory footprint is minimized, as only those parts of the operating system or utilities that are needed are linked with the application. This in turn allows companies to choose lower-cost processors in markets where the volume of application delivery could be in the millions. At such a high volume, software failure is unacceptable, and Ada is an ideal language where reliability must be guaranteed. GNAT Pro fully implements Ada and also provides multi-language support for parts of the application that may be written in C or other languages.

"GNAT Pro for Nucleus OS expands Ada's availability into industries requiring hard real-time response and very small memory footprint, such as mobile handsets, consumer electronics, or telematics/infotainment," said Neil Henderson, general manager, Embedded Systems Division, Mentor Graphics. "Ada is a well known language in the areas of safety and security. Offering Ada development support for Nucleus OS adds another layer of reliability to these small, high-volume systems that our customers develop."

"AdaCore has a long history supporting embedded operating systems in applications where safety and reliability are key concerns," said Robert Dewar, President and CEO of AdaCore. "We believe that combining the flexibility of Nucleus OS with the reliability of the Ada language will greatly benefit our customers, particularly those developing products in high-volume industries."

Pricing and Availability

The GNAT Pro for Nucleus OS is available immediately for the ARM embedded processor. Please contact AdaCore (info@adacore.com) for the latest information on pricing and supported configurations.

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.

About Nucleus Operating System

Nucleus OS was designed exclusively for real-time performance. It was conceived from the ground up for resource-constrained devices (frequency and memory) and for environments where squeezing out every cycle per watt was paramount. Thus, developers were able to focus on differentiating their products while confident their OS foundation would not adversely impact the system's overall performance.

Nucleus OS has evolved into a complete operating system composed of kernel services, extensions, and APIs. Componentizing Nucleus OS into logically related functionality not only provides significant cost savings (customers don't have to pay for what they don't use), but also provides an easy route for reducing the amount of ROM and RAM required.

About Mentor Graphics

Mentor Graphics Corporation (NASDAQ: MENT) is a world leader in electronic hardware and software design solutions, providing products, consulting services and award-winning support for the world's most successful electronics and semiconductor companies. Established in 1981, the company reported revenues over the last 12 months of about $850 million and employs approximately 4,500 people worldwide. Corporate headquarters are located at 8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777. World Wide Web site: http://www.mentor.com/.

About AdaCore

Founded in 1994, AdaCore is the leading provider of commercial software solutions for Ada, a 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 http://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, railroad 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

Nucleus is a registered trademark of Mentor Graphics Corporation. All other product and service names mentioned are the trademarks of their respective companies.

AdaCore — GNAT Pro for ELinOS

From: AdaCore Press Center
Date: Monday October 27, 2008
Subject: AdaCore and SYSGO Announce GNAT Pro for ELinOS
RSS: http://www.adacore.com/2008/10/27/elinos/

Ada now available on embedded Linux

BOSTON, October 28, 2008 — Embedded Systems Conference — AdaCore, provider of the highest quality Ada tools and support, and SYSGO, supplier of software and firmware products and services for the embedded system marketplace, today announced the availability of GNAT Pro for ELinOS™, the industrial grade embedded Linux® operating system from SYSGO. With GNAT Pro for ELinOS, embedded system developers can combine the reliability of Ada, the productivity of the GNAT Pro toolset and AdaCore support services with the configurability of ELinOS.

Because of Linux's versatility, creating an embedded Linux-based application can be a complicated process that involves selecting components, developing board support packages and drivers, and testing the whole system. ELinOS Industrial Grade Linux streamlines and automates this process, allowing developers to focus on the target applications. ELinOS incorporates appropriate tools for configuring and building the system, including a graphical configuration front-end with built-in integrity validation.

"AdaCore and SYSGO form an ideal partnership," said Jacques Brygier, Vice President of Marketing for SYSGO. "AdaCore has a well-deserved reputation for technical excellence and top-quality support, and we foresee a promising future for GNAT Pro for ELinOS to satisfy customer demand for Ada on embedded Linux."

"Building large, embedded applications is a difficult task, and developers need appropriate tools to help manage this complexity," said Robert Dewar, President and CEO of AdaCore. "ELinOS nicely complements GNAT Pro to meet these requirements, and SYSGO's corporate philosophy, with its emphasis on customer support, dovetails with ours. With the rising interest in embedded Linux, and a marketplace that simultaneously demands productivity, reliability, and efficiency, GNAT Pro for ELinOS is a winning combination."

Pricing and Availability

The GNAT Pro for ELinOS is available as an add-on to users of AdaCore's GNAT Pro development environment. Please contact AdaCore (info@adacore.com) for the latest information on pricing and supported configurations.

About ELinOS

ELinOS is a Linux based development environment designed for the creation of intelligent devices. Unlike traditional Linux implementations, SYSGO's ELinOS is purpose-built for use in demanding industrial applications. SYSGO has contributed 15+ years of field expertise to making an embedded Linux offering well suited for complex real-world applications. This deep experience also allows SYSGO to back up their Linux customers with world-class support.

About SYSGO

SYSGO excels in providing operating system technology, middleware, and software services for the real-time and embedded device market. A differentiating capability of SYSGO is the secure PikeOS™ paravirtualization operating system which is built upon a small, fast, and safe microkernel and supports the cohabitation of independent operating system personalities on a single platform, including ELinOS™, SYSGO's embedded Linux development environment. SYSGO supports international customers with services for embedded Linux, real-time capabilities and certification for safety-critical applications. Target markets include Aerospace & Defense, Industrial Automation, Automotive, Consumer

Electronics and Network Infrastructure. SYSGO customers include Airbus, Honeywell, Thales, Daimler, Raytheon, Rheinmetall, Rockwell-Collins, Siemens and Rohde & Schwarz. Today, the company has six facilities in Europe, including Germany, France and Czech Republic and offers a global distribution and support network, extending to North America and the Pacific Rim. www.sysgo.com

SYSGO, ELinOS and PikeOS are trademarks or registered trademarks of SYSGO AG in Germany and in several other countries all over the world. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. All other product and service names mentioned are the trademarks of their respective companies.

AdaCore — GPS 4.3

From: AdaCore Press Center
Date: Wednesday November 19, 2008
Subject: AdaCore Introduces Latest Version of GNAT Programming Studio
RSS: http://www.adacore.com/2008/11/19/gps-4-3/

New GPS 4.3 Integrated Development Environment enables easy configurability for multiple toolchains

NEW YORK and STUTTGART, Germany, November 19, 2008 — Automotive — Safety & Security 2008 / Ada Deutschland — AdaCore, provider of the highest quality Ada tools and support services, today announced the release of GNAT Programming Studio (GPS) 4.3 — an advanced, powerful Ada-oriented Integrated Development Environment (IDE) that accompanies the AdaCore GNAT Pro toolset on most platforms. The new release lets developers simultaneously use multiple versions of the GNAT Pro toolset and features a redesigned builder module as well as an improved documentation generator. By configuring the development environment for several toolchains, a programmer can take advantage of the latest tool improvements while continuing to use a baselined previous version of the compiler.

Among other key features, GPS 4.3 allows upgrading the IDE independent of the base compiler, so that developers can use a single IDE across multiple projects for potentially different target platforms and compilers. The redesigned builder module enables full customization, aiding development flexibility, while the improved documentation generator includes support for predefined and user-defined tags. The new features will be highlighted and demonstrated in the next GNAT Pro InSight Webinar on November 25, 2008.

"GPS continues to lead the market for professional Ada development," commented Arnaud Charlet, GPS Project Manager at AdaCore. "The requirements for GPS 4.3 were based on direct user feedback. We enhanced its ability to support multiple toolchains, and significantly updated our builder module and documentation generator to provide developers with unprecedented flexibility and control over their projects."

"Easy configurability for different toolchains is increasingly important for developers looking to use multiple GNAT Pro versions," said Robert Dewar, President and CEO of AdaCore. "With GPS 4.3, developers can use the latest GNAT Pro toolset for a variety of purposes, such as source navigation, coding standard enforcement, and metrics computation, while using a baselined, earlier compiler for actual code generation."

New functions in GPS 4.3 include:

- Easy configurability for multiple toolchains
- Redesigned and fully customizable builder module
- Improved documentation generator, including support for predefined and user defined tags
- Enhanced support for gcov (code coverage), gnatcheck (coding standard checker) and compiler switches
- Code completion with new Dynamic Mode
- Improved automatic code fixing
- New plug-ins, including:
    - o OS Shell
    - o Enhanced SPARK plug-in
    - o Show expanded code
    - o Initial support for GIT version control system

As with all GNAT Pro components, GPS is distributed with full source code and is backed by AdaCore's rapid and expert online support.

About GNAT Programming Studio (GPS)

GPS is a powerful Integrated Development Environment (IDE) written in Ada using the GtkAda toolkit. GPS' extensive source-code navigation and analysis tools can generate a broad range of useful information, including call graphs, source dependencies, project organization, and complexity metrics. It also provides support for configuration management through an interface to third-party Version Control Systems, and supports a variety of platforms, including Altix Linux, IA64 HP Linux, Solaris (sparc and x86), GNU/Linux (x86 and x86-64), Mac OS X, and x86 Windows (2003, XP and Vista). GPS is highly extensible; a simple scripting approach enables additional tool integration. It is also customizable, allowing programmers to specialize various aspects of the program's appearance in the editor for a user-specified look and feel.

Pricing and Availability

GPS 4.3 is currently available to GNAT Pro customers on selected platforms. GPS is included with the GNAT Pro Ada Development Environment. Please contact AdaCore (sales@adacore.com) for the latest information on pricing and supported configurations

About Insight Webinars

AdaCore's Insight Webinars are a series of informative webinars dedicated to tools that make up the GNAT Pro development environment. Each webinar consists of a presentation, a demo, and a question and answer session and is presented by an AdaCore technical expert.

A webinar presenting the new features of the GPS 4.3 release will take place on November 25, 2008 at 08:00am PST / 11:00 EST / 17:00 CET. For more information, or to register, please visit: http://www.adacore.com/home/gnatpro/webinars/

AdaCore — GPRbuild 1.2

From: AdaCore Releases and Updates
Date: November 18, 2008
Subject: GPRbuild 1.2.0
RSS: http://www.adacore.com/2008/11/18/gprbuild-120/

AdaCore is pleased to announce the immediate availability of GPRbuild 1.2.0 on the following host platforms (and cross builds on these platforms):

alpha-tru64
ia64-hp_linux
ia64-hpux
ia64-openvms
ia64-sgi_linux
mips-irix
pa-hpux
ppc-aix
sparc-solaris
sparc64-solaris
x86_64-linux
x86-linux
x86-lynx
x86-solaris
x86-windows

that introduces the following new features:

- Improved efficiency:
- Faster computation of recompilation needs for very large projects
- Fewer compilation artifacts generated (switch file only generated when -s is on)
- New attributes to support languages/compilers not producing objects, or not requiring a linking phase
- Improved compatibility with gnatmake

- Better support of the –subdirs option in externally built projects

AdaCore — GNAT Pro High-Integrity Edition

From: AdaCore Press Center
Date: Monday November 17, 2008
Subject: AdaCore Announces GNAT Pro High-Integrity Edition for MILS
RSS: http://www.adacore.com/2008/11/17/high-integrity-mils/

New Member of the GNAT Pro Family to Support Multiple Independent Levels of Security

NEW YORK, PARIS, and SAN DIEGO, November 17, 2008 — MILCOM 2008 — AdaCore, provider of the highest quality Ada tools and support, today announced plans to fully support the VxWorks MILS product line for medium and high security levels of the Common Criteria: Evaluation Assurance Level (EAL) 4 and above. GNAT Pro High-Integrity Edition for MILS is a full-featured Ada development environment and toolchain based on the proven GNAT Pro High-Integrity Edition for DO-178B. It includes the Cert run-time library that is part of numerous safety-certified systems and which can meet Common Criteria Security Assurance Requirements equivalent to EAL 4.

The Ada language has a long history of satisfying requirements for developing safe and secure systems and is in use in many flight-critical avionics applications. Several of these have been certified to DO-178B Level A, the highest criticality level. AdaCore's GNAT Pro High-Integrity Edition for DO-178B run-time library has been or will be certified to DO-178B Level A in multiple avionics systems, including those on the Boeing 787, C-130AMP and KC-767. Industry studies have shown that this certification evidence satisfies EAL 4 security requirements.

In the case of high security systems, EALs above 4 require semi-formal or formal methods depending on the level. AdaCore has entered into a strategic partnership with Praxis High Integrity Systems to meet this goal. Praxis, along with the NSA and AdaCore, recently announced the release of the Tokeneer project for public availability. Tokeneer is a research project funded by the NSA that used the Praxis Correctness by Construction methodology and the Ada-based SPARK language to build a high security application. The project was completed ahead of schedule and within budget. The final product was developed to be able to meet EAL 5. The partnership between AdaCore and Praxis will see the integration of the SPARK technology with the GNAT Pro High Integrity Edition for MILS product line to offer tools to meet these higher levels of security. See: http://www.adacore.com/pr/tokeneer

In addition AdaCore has entered into partnerships with leading class vendors to provide a total package to satisfy high security application development needs. At the core of this partnership is Wind River with its upcoming VxWorks MILS Platform 2.0. This platform is currently undergoing security certification and will be certified to the Common Criteria for the Separation Kernel Protection Profile, which requires EAL 6+. For communication within and across partitions, AdaCore has partnerships with both RTI and OIS that offer off-the-shelf products supporting middle-ware DDS or CORBA based communications for the MILS architecture. These components allow developers to meet security levels of EAL 4 or higher.

"AdaCore is providing a unique addition to our VxWorks MILS platform," said Marc Brown, Vice President of VxWorks Marketing at Wind River. "The VxWorks MILS platform will provide the secure underlying architecture required to meet top levels of security. The GNAT Pro High-Integrity Edition for MILS provides a tool set to help developers meet these stringent security requirements for their application development."

"AdaCore has a long history supporting embedded application projects where safety and reliability are key concerns," said Robert Dewar, President and CEO of AdaCore. "We believe our past experience with safety-critical systems provides a natural basis for supporting high-security application development. The combination of our own expertise with our partners makes this new product unique in the industry for security application development."

"Praxis and AdaCore have been partners for several years," said Rod Chapman, Principal Engineer with Praxis High Integrity Systems Ltd. "The GNAT Pro tool sets have been successfully used on a range of systems, including those with stringent safety and security requirements. This partnership allows Praxis to lend its security expertise, SPARK language and tool set as part of a joint package to meet high-level security development project requirements."

AdaCore — GNAT Pro for VxWorks SMP Capability

From: AdaCore Press Center
Date: Monday November 17, 2008
Subject: AdaCore Announces GNAT Pro for the VxWorks SMP Capability
RSS: http://www.adacore.com/2008/11/17/vxworks-smp/

Bringing a multi-threaded language to multi-core

NEW YORK, PARIS, and SAN DIEGO — MILCOM 2008 — AdaCore, provider of the highest quality Ada tools and support, today announced the availability of GNAT Pro for Wind River's VxWorks 6.6 SMP product. This powerful Ada development environment brings a language that was designed from the ground up to support multi-processing to an operating system that takes advantage of multi-core processors.

Multi-core technology is the next transformative technology for the Device Software Optimization (DSO) industry. With the SMP add-on product installed, VxWorks platforms are enhanced with symmetric multi-processing (SMP) capabilities within the operating system, network stack, and development tools to provide the easiest path to realize the benefits of multi-core technology.

The Ada language is uniquely positioned to take advantage of this new technology. Ada supports multi-processing via its tasking construct. Multi-threading, mutual exclusion, and inter-process communication are handled by the rendezvous mechanism and protected types/objects. The latter provides a reliable and efficient building block for defining semaphores and events for inter-task communication.

"GNAT Pro brings a powerful programming language to our new multi-core platform," said Rob Hoffman, Vice President and General Manager of Aerospace and Defense at Wind River. "This provides a unique capability for VxWorks users, especially those developing applications that need to meet high-reliability requirements. Users can take existing Ada software and port it to our SMP platform to immediately take advantage of its multi-processing capabilities. Moreover, new programs can also instantly take advantage of this capability when developing Ada or mixed-language applications."

"This is an exciting time for the software industry," said Robert Dewar, President and CEO of AdaCore. "The Ada language was designed from the start for multi-threaded applications. On a single cpu system this was typically handled by a multiplexing scheduler to simulate parallel processing. With multi-cpu's or multi-core processors these tasks may now truly run in parallel physically. While developers using other programming languages are struggling to find how to take advantage of this new hardware, Ada developers can immediately leverage tried and tested features that have been in the language for years."

AdaLog — AdaControl

From: Jean-Pierre Rosen <rosen@adalog.fr>

Date: Tue, 21 Oct 2008 18:02:01 +0200
Subject: AdaControl 1.10r8 released
Newsgroups: comp.lang.ada

Adalog is pleased to announce the release of a new version of AdaControl, now reaching 372 possible checks! Plus many improvements to other rules and usability (even some bug fixes).

The executable distributions are now pre-compiled for GNAT/GPL2008

In addition, Windows users will appreciate that both the source and executable versions are also distributed through an installer that will take care of everything. (Thanks to Inno-Setup, a great free installer generator).

As usual, everything is under the GMGPL. Commercial support and services are available from Adalog.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 14 Nov 2008 17:17:13 +0100
Subject: AdaControl 1.10r10 released
Newsgroups: comp.lang.ada

This is just a bug fix release, fixing two problems in case of complicated renamings. No changes in functionality.

Please update, sorry for the inconvenience.

Aonix — Certification Kit for ObjectAda RAVEN

From: Aonix Press Release
Date: October 14, 2008
Subject: Aonix Brings Ada Kernel Certification Kit to Market
URL: http://www.aonix.com/pr_10.14.08.html

New DO-178B Certification evidence available for ObjectAda® RAVEN™

San Diego, CA, October 14, 2008 — Aonix®, a provider of solutions for safety- and mission-critical applications, announced the release of a new certification kit for ObjectAda RAVEN. Built in response to requirements from existing customers, Aonix has designed the ObjectAda RAVEN certification kit to provide all the evidence and tools required to assist customers for certification to DO-178B Level A as well as to other standards.

ObjectAda RAVEN enables engineers to build applications for deployment in safety-critical applications such as those found in transportation, avionics and flight systems, and nuclear energy management where stringent standards must be followed and proof of conformance is essential to obtain certification from associated authorities. The ability to procure COTS certification evidence together with ObjectAda RAVEN can yield dramatic savings where costs for production of certification evidence can soar from $50 to hundreds of dollars per source line of code.

"By providing developers with products that improve code reliability, reduce testing overhead, and shorten certification cycles, Aonix helps customers realize significant savings and furthers its reputation for facilitating the certification process," noted Gary Cato, director of marketing at Aonix. "The ObjectAda RAVEN certification kit substantially reduces the engineering hours, which—given the cost of producing certification evidence—can be a critical lifeline for a project."

ObjectAda RAVEN for Windows consists of an Ada 95 compiler with the supporting tools of a build/bind tool, library tool and debugger, and is delivered with a predefined program library which conform to the Ravenscar profile subset of the full Ada language. The Ravenscar profile, adopted at the Eight International Real-Time Ada Workshop (IRTAW-8), Ravenscar UK, and subsequently made part of the Ada 2005 specification, accommodates certification requirements for high-integrity, safety-critical, real-time systems. ObjectAda RAVEN allows developers to choose between the traditional Aonix IDE for development and the optional AonixADT™ Eclipse plug-in. Geared to maximize developer ease and efficiency, AonixADT incorporates Ada-project awareness, an Ada-language sensitive editor, Ada-language compile and build capabilities, and a complete Ada debugger interface, ensuring that Ada developers enjoy state-of-the-art interface capabilities.

Shipping and Availability

ObjectAda Real-Time RAVEN for Windows targeting PowerPC is immediately available. Prices range from $15,000 to $25,000 for a single seat license depending on bundling options. Quantity discounts for development licenses are available. Certification is available now with pricing based on project-specific requirements.

Aonix — ObjectAda Real-Time for LynxOS

From: Aonix Press Release
Date: October 1, 2008
Subject: Aonix Enhances Ada Real-Time Tools for LynxOS RTOS
URL: http://www.aonix.com/pr_10.01.08b.html

New ObjectAda Real-Time efficiencies reduce development cycles

Embedded Systems Show, Birmingham, UK, October 1, 2008 — Aonix®, a provider of solutions for safety- and mission-critical applications, announced the release of ObjectAda for Linux hosts targeting PowerPC embedded and real-time systems running the LynuxWorks LynxOS 4.2 RTOS. With enhanced edit, build, and debug facilities, ObjectAda speeds Ada application development using Linux, the preferred environment of engineers building large and complex systems. Thanks to such time savings, ObjectAda developers can reduce time to market and development costs over less functional toolchains.

ObjectAda Real-Time for LynxOS includes an enhanced linker that dramatically reduces link time. As well, the Ada debug facility enables more efficient use of functions to step into and out of protected subprograms. Upgraded support for GDB/MI mixed-code disassembly and interspersion of disassembled code with sources is included and improved multi-language debug support for C and Ada code are also provided to speed cross language test cycles.

"Aonix has long been a premier partner," said Joe Wlad, Director of Certification, Marketing and Services at LynuxWorks. "Many of our customers, especially in the aerospace and military segments, continue to look for ways to leverage their Ada development expertise in new projects or extend the life of their existing Ada applications with updated and improved tools. We are pleased that Aonix continues to support our most important LynxOS customers with their ObjectAda family."

"Access to standards-based COTS products like those supplied by LynuxWorks is extremely important for our military and defense customers," confirmed Gary Cato, Aonix director of marketing. "Aonix is committed to supporting world-class RTOS environments like LynxOS and strives to continually improve the efficiency and level of integration between our products. Our combined technologies deliver a powerful toolset to developers."

About the ObjectAda Family

ObjectAda is an extensive family of native and cross development tools and runtime environments. ObjectAda native products provide host development and execution support for the most popular environments including Windows, Linux and various Unix operating systems. ObjectAda Real-Time products provide cross development tools on Windows, Linux or Unix systems which target PowerPC and Intel target processors running in a full Ada "bare" runtime or in conjunction with popular RTOSs. ObjectAda RAVEN® products provide a hard real-time Ada runtime to address those systems requiring certification to the highest levels of safety standards such as DO-178B Level A for flight safety.

About LynxOS

LynxOS is the leading POSIX conformant embedded operating system on the market—the first choice of customers who cannot afford downtime. LynxOS forms the embedded core of a wide array

of systems, from postal mail sorting solutions to air traffic control systems, and from office printers and copiers to commercial airliners. As the only hard-real time technology with broad conformance to open and de facto standards such as Linux®, POSIX and UNIX®, LynxOS has the power to take full advantage of current hardware designs, and offers broad re-use of available software applications. LynxOS also features a fully integrated Eclipse-based IDE supporting the ubiquitous GNU toolchain to provide a streamlined development and execution environment and offers a unique Linux Application Interface (ABI) for running Linux applications in high-reliability systems.

Shipping and Availability

ObjectAda Real-Time targeting Power Architectures running LynuxWorks' LynxOS 4.2 is immediately available starting at $15K in the U.S. with quantity discounts available. Support for LynxOS 5.0 will be determined based on customer requirements.

About LynuxWorks

LynuxWorks, a world leader in the embedded software market, is committed to providing open and reliable real-time operating systems (RTOS) and software tools to embedded developers. The company's LynxOS family of operating systems offers open standards with the highest level of safety and security features, enabling many mission-critical systems in defense, avionics and other industries. Additionally, LynuxWorks' BlueCat Linux provides the features and support of embedded Linux for companies wanting to use open source technology for their embedded applications. The Eclipse-based Luminosity IDE gives a powerful and consistent development system across all LynuxWorks operating systems. Since it was established in 1988, LynuxWorks has created technology that has been successfully deployed in thousands of designs and millions of products made by leading communications, avionics, aerospace/defense, and consumer electronics companies. LynuxWorks' headquarters are located in San José, CA. For more information, visit www.lynuxworks.com.

Ada and GNU/Linux

Ada Designer and GNATGPR Debian packages

From: David Sauvage <pariakaman@gmail.com>
Date: Sat, 13 Sep 2008 05:49:49 −0700 (PDT)
Subject: Debian repository for AdaDesigner & GNATGPR packages
Newsgroups: comp.lang.ada

AdaDesigner [1] and GNATGPR [2] are now available as Debian packages in my launchpad personal package archive [3] for i386, amd-64 and lpia.

They should work on Debian Lenny and Ubuntu Intrepid. Please note they are not yet fully compliant with the Debian package policy.

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

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

[3] deb http://ppa.launchpad.net/pariakanet/ubuntu intrepid main

deb-src http://ppa.launchpad.net/pariakanet/ubuntu intrepid main

adadesigner
libgnatgpr0
libgnatgpr0-bin
libgnatgpr0-dev

Debian Ada Policy

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Sat, 1 Nov 2008 10:57:36 −0700 (PDT)
Subject: ANNOUNCE: Debian Ada Policy, Third Edition for Debian 5.0 "Lenny"
Newsgroups: comp.lang.ada

I have updated the Debian Ada Policy to reflect how the packages are now done in Debian 5.0 "Lenny". This document is available in HTML, text, PDF, info and texinfo format at

http://www.ada-france.org/debian/debian-ada-policy.html,

http://www.ada-france.org/debian/debian-ada-policy.txt,

http://www.ada-france.org/debian/debian-ada-policy.pdf,

http://www.ada-france.org/debian/debian-ada-policy.info,

http://www.ada-france.org/debian/debian-ada-policy.texi.

The texinfo source of this document is available in Ada-France's public monotone repository[1].

[1] http://www.ada-france.org/article131.html

Summary of changes since the Second Edition for Debian 4.0 "Etch":

- Amendment 1 (aka Ada 2005) published
- Transition to GCC 4.3
- Move to library project files, both for building and using libraries
- GLADE superseded by PolyORB (but not provided in Debian, unfortunately)
- Libraries now provide debugging information in a new -dbg package

References to Publications

AdaCore — Article on the Couverture Project

From: AdaCore Developer Center
Date: Tuesday September 9, 2008
Subject: Boosting coverage
RSS: http://www.adacore.com/2008/09/09/boosting-coverage-2/

In an article, recently published in New Electronics entitled "Boosting coverage", Matteo Bordin looks at how process and new virtualization technology, being developed under the Coverage Project, will be used to ease the burden of carrying out effective coverage analysis on code.

To read the article, please click here or visit:

http://fplreflib.findlay.co.uk/articles/15269/Boosting%20coverage.pdf

CrossTalk — "Safety and Security: Certification Issues and Technologies"

From: AdaCore Press Center
Date: Wednesday October 1, 2008
Subject: Safety and Security: Certification Issues and Technologies
RSS: http://www.adacore.com/2008/10/01/safety-and-security-certification-issues-and-technologies/

AdaCore — Nov 2008 Contents

From: AdaCore Press Center
Date: Monday November 10, 2008
Subject: Nov 2008 Contents
RSS: http://www.adacore.com/2008/11/10/nov-2008-contents/

Nov 2008 Contents

- NSA-Sponsored Tokeneer Software Available
- Project Coverage Launch Initiates
- Open AdaCore Series
- New Platforms
- In the Pipeline
- Academia Corner: Mälardalen University
- Interview with Ed Falis
- Webinar Schedule
- Traceability Analysis Expands Safety-Critical Ada
- Conferences/Events

[See http://www.adacore.com/wp-content/uploads/2008/11/adacore_news_1008_web.pdf —su]

AdaCore — Automotive 08 / Ada Deutschland

From: AdaCore Press Center
Date: Wednesday September 17, 2008
Subject: Automotive 08 / Ada Deutschland
RSS: http://www.adacore.com/2008/09/17/automotive-08-ada-deutschland/

AdaCore will be exhibiting at this event and is also sponsor of the gala dinner.

Rapita — Tutorial at ESS'08

From: Rapita Systems News
Date: 15 September, 2008
Subject: See Guillem Bernat's tutorial at ESS'08 entitled "Timing Matters: Optimising the worst-case execution time"
RSS: http://www.rapitasystems.com/node/344/

Rapita Systems' CEO, Dr. Guillem Bernat, will be giving a technical presentation entitled, "Timing matters: Optimising the worst-case execution time."

"How long does my software take to run…and how can I reduce it?" are key issues for embedded engineers designing reliable systems. Understanding, verifying, and improving the timing performance of their real-time products gives successful companies in the avionics, telecommunications, space, and automotive electronics industries a key competitive edge.

Today, software timing analysis doesn't have to be guesswork. This presentation addresses two key aspects of real-time systems performance:

1. How to gain a clear, detailed, and accurate understanding of the execution time behaviour of real-time embedded software.
2. How to target optimisation effort precisely where it will have the maximum benefit in improving system timing behaviour, for the minimum cost.

This presentation will cover key aspects of real-time systems: finding worst case execution times (WCET) and the worst case path, and why worst case optimization is not the same as average execution optimization. Practical issues of looking for the worst-case "hot-spots", identifying timing bugs and verifying optimisation opportunities will be explained with examples of worst case optimizations.

Further details of the conference can be found on the Embedded Systems Show website.

AdaCore — Talk at FOSDEM 2009

From: AdaCore Press Center
Date: Friday December 5, 2008
Subject: FOSDEM 2009
RSS: http://www.adacore.com/2008/12/05/fosdem-2009/

Thomas Quinot will be giving a talk on "Ada Annex E — Distributed Systems".

Vincent Celier will be giving talks on "GPS — The GNAT Programming Studio" and "GNATBench — Ada programming with Eclipse".

Ada Inside

AdaCore — NSA Releases Secure Software Project

From: AdaCore Press Center
Date: Monday October 6, 2008
Subject: NSA Releases Secure Software Project to Open Source Community
RSS: http://www.adacore.com/2008/10/06/nsa-releases-secure-software-project-to-open-source-community/

Tokeneer project shows the way to develop secure systems in a rigorous and cost-effective manner

NEW YORK, PARIS, and BATH, U.K., October 6, 2008, — VSTTE 2008 — The development of highly secure, low defect software will be dramatically helped by the release of the Tokeneer research project to the open source community by the US National Security Agency (NSA). The project materials, including requirements, security target, specifications, designs, source code, and proofs are now available at www.adacore.com/tokeneer.

The Tokeneer project was commissioned by the NSA from UK-based Praxis High Integrity Systems as a demonstrator of high-assurance software engineering. Developed using Praxis' Correctness by Construction (CbyC) methodology it uses the SPARK Ada language and AdaCore's GNAT Pro environment. The project has demonstrated how to meet or exceed Evaluation Assurance Level (EAL) 5 in the Common Criteria thus demonstrating a path towards the highest levels of security assurance.

The unprecedented release of the project into the open source community aims to demonstrate how highly secure software can be developed cost-effectively, improving industrial practice and providing a starting point for teaching and academic research. Originally showcased in a conference paper in 2006, it has the long-term aim of improving the development practices of NSA's contractors. Tokeneer was created as a fixed-price project, taking just 260 person days to create nearly 10,000 lines of high-assurance code, achieving lower development costs than traditional methods per line of code.

"The Tokeneer project has the potential to revolutionize the development of highly secure systems," said Robert Dewar, President and CEO of AdaCore. "By releasing Tokeneer to the community, the NSA will help drive good programming practice and demonstrate the importance of SPARK and Ada to the emerging security market. We are delighted to be involved with Praxis and the NSA in this ground-breaking project."

Tokeneer has been written in SPARK Ada, a high level programming language designed for high-assurance applications. Originally a subset of the Ada language, it is designed in such a way that all SPARK programs are legal Ada programs. Ada is the natural choice for mission-critical, high-integrity systems due to its combination of flexibility, reliability and ease of use, and SPARK further adds a static verification toolset that combines depth, soundness, efficiency and formal guarantees.

"We are extremely proud of the Tokeneer project," said Keith Williams, Praxis Managing Director. "We hope the research, teaching, and open-source communities will put the material to good use as a model of high-assurance software development."

The project is aimed at both the industrial and academic communities, forming an ideal base for further research in program verification and as a high level teaching aid for educators. It will also be contributed to the Verified Software Repository under the auspices of the current "Grand Challenge" in Dependable Systems Evolution.

"The Tokeneer project is a milestone in the transfer of program verification technology into industrial application. Publication of the full documents for the project has provided unprecedented experimental material for yet further development of the technology by pure academic research. It will serve as a touchstone to chart and measure progress of the basic science of programming, on which the technology is based."

Sir Tony Hoare, Fellow of the Royal Society (FRS) of Microsoft Research, and founder of the Grand Challenge

"The publication by Praxis and NSA of the Tokeneer system is a fantastic contribution to the software engineering research and teaching community. Good case studies have been very hard to find, and have often been proprietary. Finally, we have a full and open example of a development from a world leader in high integrity systems with exemplary requirements, specifications, design and code. I'm very excited about the impact this might have in our field, in both teaching and research, and the potential it might have in moving us towards a more open community with greater collaboration between industry and

academia, and a more constructive engagement of theory and practice."

Professor Daniel Jackson of Massachusetts Institute of Technology, Computer Science Laboratory

"Ada is a very complete language that allows us to present procedural and object-oriented concepts. It supports and encourages data abstraction and provides helpful diagnostic messages to beginning students when they make the usual kinds of beginner's mistakes. In addition, Ada programs are quite readable due to well-chosen syntax and recognizable symbols for standard operations, and once a student has learned Ada, he or she finds it easy to learn a second language, such as Java or Visual Basic."

About Praxis and Correctness by Construction

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 and Nuclear. The company operates internationally with active projects in the US, Asia and Europe. The UK Headquarters are in Bath with offices also in London, Loughborough and Paris. It is wholly owned by Altran Technologies which is a global leader in innovation engineering and employs 17,500 engineers across the world. www.praxis-his.com

Correctness by Construction (CbyC) is Praxis's method of developing software. CbyC uses tools and techniques that aim to make it both difficult to introduce defects during software development, and straightforward to correct defects early in the development lifecycle. These tools and techniques are often required by industry standards for safety and security critical software, and as a result Praxis has developed an expert capability and track record for the development of such software. CbyC is cost effective because reducing defects significantly reduces risk and rework.

About National Security Agency

The National Security Agency/Central Security Service is America's cryptologic organization. Further information is available from the NSA website. www.nsa.gov

AdaCore — Tokeneer Project Material Available

From: AdaCore Developer Center
Date: Monday October 6, 2008
Subject: Tokeneer research project available for download
RSS: http://www.adacore.com/2008/10/06/tokeneer-research-project-available-for-download/

The development of highly secure, low defect software has been dramatically helped by the release of the Tokeneer research project to the open source community by the US National Security Agency (NSA). The project materials, including requirements, security target, specifications, designs, source code, and proofs are now available at www.adacore.com/tokeneer."

The Tokeneer project was commissioned by the NSA from UK-based Praxis High Integrity Systems as a demonstrator of high-assurance software engineering. Developed using Praxis' Correctness by Construction (CbyC) methodology it uses the SPARK Ada language and AdaCore's GNAT Pro environment. The project has demonstrated how to meet or exceed Evaluation Assurance Level (EAL) 5 in the Common Criteria thus demonstrating a path towards the highest levels of security assurance.

Download Tokeneer »
http://www.adacore.com/tokeneer

Ada in Context

Generic_Roots

From: John B. Matthews <jmatthews@wright.edu>
Date: Sat, 01 Nov 2008 13:47:16 −0400
Subject: Generic_Roots
Newsgroups: comp.lang.ada

Although they never became standard, I've written implementations of

AI-356: Ada.Numerics.Generic_Real_Arrays.Generic_Roots

AI-356: Ada.Numerics.Generic_Complex_Arrays.Generic_Roots

The updated numerics packages and some test code are available here

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

The implementation uses the Durand-Kerner-Weierstrass method:

http://en.wikipedia.org/wiki/Durand-Kerner_method

I'd be grateful for any comments you might have.

From: John B. Matthews <jmatthews@wright.edu>
Date: Fri, 07 Nov 2008 08:36:44 −0500
Subject: Re: Generic_Roots
Newsgroups: comp.lang.ada

> Umm, AI95-0356 is "Support for Preemption Level Locking Policy".

You mean AI95-0346 "Roots of polynomials".

Yes, thank you! I've corrected the reference and added links to the CVS.

> You might want to pass this implementation past John Barnes, as he was the original proposer of that AI.

Ah, I see now that he was the initial author. In the implementation advice, he suggested "using established techniques such as Laguerre's method." My implementation is definitely experimental, but it's a convenient alternative. My test cases are limited, so I'd be interested to hear of any anomalies. Of course, I expect most users are using BLAS.

Offset added to 'Address

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 19 Sep 2008 01:09:53 −0500
Subject: Re: Adding offset to 'Address
Newsgroups: comp.lang.ada

>>> Raw_Bytes : access Unsigned_Char

>>> for Ethernet_Header'Address use Raw_Bytes.all'Address;

>>> and additionally need something like:

>>> for IP_Packet'Address use Raw_Bytes.all'Address + Ethernet_Header'Size;

>> It's not clear to me exactly what problem you are trying to get past.

> The problem was basically to perform arithmetics on system addresses

No, that's a (bad) solution to some problem, not a problem by itself. The problem seems to be how to access the data bytes without accessing the header, and Tom suggests a good solution for that. (We use versions of it in Claw.)

IMHO, the use of Address in Ada 95 or newer code (other than to specify raw, absolute machine addresses) is never necessary and represents bad programming. I see it fairly often here, and it makes me ill. I much prefer general access types and possibly Unchecked_Conversions, because you lose much less strong typing that way (and the code is better without the need for heroic efforts on the part of the compiler).

On the mapping of Ada Tasks to OS threads

From: "Peter C. Chapin" <pcc482719@gmail.com>
Date: Sat, 27 Sep 2008 19:14:05 −0400
Subject: Re: Blocking syscalls in Tasks
Newsgroups: comp.lang.ada

> My understanding is the same — or at least I nowhere found something else But I wonder about this, because in my opinion there is a great difference

between tasks = userspace threads and tasks = kernelspace threads.

I think the reason this is probably left implementation defined is so that Ada can be implemented on systems that don't provide kernel threads to the applications. On such systems, the Ada runtime can implement user mode threads and still obey the standard.

User mode threads do have some nice properties on their own however… specifically, fast context switching times. Thus even when kernel threads are available an implementation might want to provide some options in this area.

> Hence, I think there should at least be some way to check whether in the given implementation of tasks blocking syscalls block only the calling task or everything.

One imagines it would be documented some place. At least you would like to believe that.

From: Schwering <schwering@gmail.com>
Date: Sat, 27 Sep 2008 13:02:34 −0700 (PDT)
Subject: Re: Blocking syscalls in Tasks
Newsgroups: comp.lang.ada

> I've tried to check for a reliable references, and found something which may help you to make some assertions. It seems that it is thread/ task blocking rather than process blocking. This is based on mail exchanged by developers of the Linux kernel. It is dated Sun, 12 Aug 2007, so do not make assertion using this about too much old kernels (you did not say what kernel version you use, neither if it is a vanilla one or not)

Here is the link to the mail at the LKML mail archives: http://lkml.org/lkml/2007/8/12/102

[…]

I think this only refers to the Linux kernel's threads. As far as I know (I'm far from being an expert in threads..), in kernelspace threads, a syscall does not block the entire process but only the calling thread. In fact, this is the main advantage over userspace threads.

I don't know whether Ada tasks necessarily are threads and whether it is specified whether they are kernel or userspace threads or some hybrid stuff.

From: Maciej Sobczak <see.my.homepage@gmail.com>
Date: Sat, 27 Sep 2008 14:54:48 −0700 (PDT)
Subject: Re: Blocking syscalls in Tasks
Newsgroups: comp.lang.ada

> Are tasks commonly implemented using threads

You should expect that from the quality implementation. This not only gives some nice properties for potentially blocking I/O calls (exactly — that only the calling task gets blocked), but also allows to safely use system-level synchronization primitives (like mutexes) for interactions between tasks.

Not that this would be recommended over standard Ada mechanisms, but it might actually happen even behind the scenes when using some external library. Knowing that Ada tasks are just system threads in disguise allows to use such libraries safely. Without this guarantee the interoperability of Ada would be limited.

On the other hand, even this assumption (that tasks are threads in disguise) is not enough to assert that Ada code can be safely executed by threads that do not originate from or at least where not prepared by the Ada runtime, although in a quality implementation this might be a very welcome property.

I don't know whether GNAT provides this property.

From: Tom Moran <tmoran@acm.org>
Date: Sun, 28 Sep 2008 12:48:22 −0500
Subject: Re: Blocking syscalls in Tasks
Newsgroups: comp.lang.ada

> Except for DOS, in current OS all Ada partitions are executed by native OS threads.

This is confusing. An Ada partition is not at all the same as an Ada task, and there's no guarantee that an Ada task maps 1-1 to an OS thread.

> Now, in a GUI system, the RC_TASK (resource task) is blocked until a input device such as the mouse or keyboard actives the thread.

MS Windows tasks do not block waiting for mouse or keyboard. Windows "event based" GUI design was based on a single thread and a polling loop.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Mon, 29 Sep 2008 21:35:52 −0500
Subject: Re: Blocking syscalls in Tasks
Newsgroups: comp.lang.ada

Probably because of Janus/Ada, pretty much everything you said about the mapping of Ada tasks to threads is wrong.

Janus/Ada still maps all tasks to one Windows thread. That was originally supposed to be a temporary Q&D implementation, but for a variety of reasons it never got replaced. Most obviously: other things needed work more urgently than the tasking, which is quite efficient. Depending on your circumstances, it might actually be faster than a threaded implementation. (Our ultimate goal is to have both.)

In any case, the point is that Ada doesn't say anything about the mapping of tasks to OS threads: you simply have to ask your vendor. And, of course blocking of system calls follows from that.

Compilers supporting Ada 2005

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Mon, 06 Oct 2008 19:12:54 +0200
Subject: Ada compilers supporting Ada 2005 (was: Blocking syscalls in Tasks)
Newsgroups: comp.lang.ada

> Most people want to know about Ada 2005 not Ada 95, and the one and only vendor that supports the Ada 2005 is AdaCore with GNAT. The information that I gave was based on that vendor's system.

More on which compilers do support Ada 2005:

"Most recently, Mr. Baird worked for IBM, which acquired Rational in 2003. As Senior Software Engineer, he was responsible for adapting the Ada 95 middle pass portion of the Rational Ada compiler to implement the dynamic semantics of Ada 2005."

So IBM is also the only vendor supplying a compiler for Ada 2005.

(Quoted from AdaCore's press center announcing "the appointment of Ada expert Stephen Baird to the company's GNAT Pro implementation team", 2008-09-09.)

From: Colin Paul Gloster <Colin_Paul_Gloster@ACM.org>
Date: Tue, 7 Oct 2008 12:51:38 +0100
Subject: Re: Ada compilers supporting Ada 2005 (was: Blocking syscalls in Tasks)
Newsgroups: comp.lang.ada

[…]

I do not believe that an Ada 2005 compiler is available yet from IBM.

Samuel F Scheerens of International Business Machines sent by email on July 2nd, 2008:

"[..]

> Why do you still not sell an Ada 2005 compiler?

We are working on support for Ada 2005. Jim can give you details, but you may have to sign some sort of non-disclosure agreement to get specifics. [..]"

Jim of I.B.M. sent by email on July 3rd, 2008:

> "[..]

Ada 2005 will be available later this year. We have a working version of the compiler, but not quite all of the container library. Our customers haven't been pushing for early release. [..]"

From: Britt Snodgrass <britt.snodgrass@gmail.com>
Date: Tue, 7 Oct 2008 07:31:42 −0700 (PDT)

Subject: Re: Ada compilers supporting Ada 2005 (was: Blocking syscalls in Tasks)
Newsgroups: comp.lang.ada

> Is a genuine Ada 2005 compiler available from AdaCore?

What do you mean by "genuine"? AdaCore has been far, far more proactive with Ada 2005 support than any other vendor. Other Ada compiler vendors are seemingly moribund, a sad situation. Rational Apex used to be a great product, but IBM has let it atrophy, making absolutely no effort to be competitive or market it to new customers.

[…]

Also, there is an old (April, 2008) announcement on http://www-01.ibm.com/support/docview.wss?uid=3Dswg21221323. But unless IBM changes their business model for Apex, it will come too late to be competitive.

From: Colin Paul Gloster <Colin_Paul_Gloster@ACM.org>
Date: Tue, 7 Oct 2008 16:39:55 +0100
Subject: Re: Ada compilers supporting Ada 2005 (was: Blocking syscalls in Tasks)
Newsgroups: comp.lang.ada

>I'm interested to know which vendors are serious about supporting the standard.

I think it is clear that RRSoftware is serious about this, even if it does not have a suitable product available yet. Similarly for Irvine, judging from answers to queries I asked in Summer 2008.

Sofcheck also seems to be serious but paying clients have been demanding other things instead of Ada 2005. Of course, if you wish to pay for it…

[…]

I was told by IBM in 2008 that because Apex is not available alone by itself that I would need to acquire many other items — none of which I need, such as a compiler for another language and a different version tracking system than what many people I collaborate with already use — from IBM for a lot more money than from some of the other compilers.

Exceptions on NAN with GNAT

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Thu, 20 Nov 2008 14:46:12 +0100
Subject: Re: Forcing exceptions on NANs with GNAT?
Newsgroups: comp.lang.ada

> is it possible to influence the behaviour of GNAT regarding the handling of NANs? (Most importantly in the special case of division by zero.)

We need to get exceptions whenever a NAN is generated, is this possible somehow? (For example by setting Machine_Overflow to True and recompiling the compiler itself.)

You can scrap IEEE stuff in favor of Ada semantics by declaring your own floating-point [sub]type with a range specified. The compiler will be forced to check values:

```
type Safe_Float is digits 6
   range -10.0E10..+10.0E10;
```

or

```
subtype Safe_Float is Float
   range Float'Range;
```

then

```
   X : Safe_Float := 1.0;
   Y : Safe_Float := 0.0;
begin
   Y := X / Y;
exception
   when Error : others =>
-- Should  print "range check failed"
   Put_Line (
         Exception_Message (Error));
end;
```

From: Markus Schoepflin <nospam@no.spam>
Date: Fri, 21 Nov 2008 10:33:27 +0100
Subject: Re: Forcing exceptions on NANs with GNAT?
Newsgroups: comp.lang.ada

[…]

The suggestion from Dmitry (create a subtype with the same range as float) works fine for my purposes, GNAT then raises range check errors for both division by zero and overflows.

C macros and their equivalent in Ada

From: Ray Blaak <rblaa@telus.net>
Date: Tue, 09 Sep 2008 16:33:10 GMT
Subject: Re: C Macros and their equivalent in Ada
Newsgroups: comp.lang.ada

> are there Ada equivalents of the C macros like: __DATE__, __TIME__

> How about the function Ada.Calendar.Clock to get the current instant

The idea is to get the date/time the file was compiled.

Rasika, what I have often done is generate a source file from my build script that has the version and build dates baked in. E.g. using ant:

```
<property name="VERSION"
  value="1.1"/>
<target name="generate.version">
  <tstamp>
    <format property="yyyyMMdd"
      pattern="yyyyMMdd"/>
  </tstamp>
    <echo file=
      "src/Support/Version.adb">
package body Version is
  function Version return String is
  begin
    return "${VERSION}";
  end;
  function Build return String is
  begin
    return "${yyyyMMdd}";
  end;
end;
    </echo>
  </target>
```

[…]

From: Ray Blaak <rblaa@telus.net>
Date: Wed, 10 Sep 2008 00:16:58 GMT
Subject: Re: C Macros and their equivalent in Ada
Newsgroups: comp.lang.ada

I would argue that generating the version file explicitly from a build script is actually superior to __DATE__/__TIME__ macros, since the actual compilation build time is often inconsistent.

If the version file has not actually changed in a while, for example, typical build environments just use the last object file, meaning the datestamp can be quite obsolete and thus inaccurate.

By tying it to the build script, you can do things like saying "make an official release NOW", and know that the version datestamp is tied to that explicit action.

This is important to allow one to correlate the datestamp to changelogs, release notes, etc.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 9 Sep 2008 07:39:28 −0700 (PDT)
Subject: Re: C Macros and their equivalent in Ada
Newsgroups: comp.lang.ada

No, Ada does not have macros. I am not aware of any compiler that provides the equivalent of C's __DATE__ and __TIME__ macros (which, for those here who don't know C, expand to the date and time at which preprocessing took place, as strings).

If you need that feature, you're better off generating an Ada source file from the "date" utility, e.g.

```
echo "package Compilation_Date is" >
  compilation_date.ads
echo "   Timestamp : constant String :=
  \"$(date --full-ISO)\";" >>
  compilation_date.ads
echo "end Compilation_Date;" >>
  compilation_date.ads
```

From: Adam Beneschan <adam@irvine.com>
Date: Tue, 9 Sep 2008 08:11:24 -0700 (PDT)
Subject: Re: C Macros and their equivalent in Ada
Newsgroups: comp.lang.ada

Ada doesn't have a standard mechanism for providing the compilation date and time. There's no reason why a particular implementation couldn't provide something like this, though—perhaps a package with special String constants or functions (most likely using pragma Import) that would cause the compiler to substitute the compile date/ time when the constant or function is used. Our compiler has something like this. Don't know whether GNAT does. If you need it, you may want to ask your compiler vendor for this sort of feature. It shouldn't be terribly difficult to implement.

From: Maciej Sobczak <maciej@msobczak.com>
Date: Tue, 9 Sep 2008 13:57:11 −0700 (PDT)
Subject: Re: C Macros and their equivalent in Ada
Newsgroups: comp.lang.ada

[…]

```
> echo "package Compilation_Date is" >
   compilation_date.ads
  echo "  Timestamp : constant String :=
   \"$(date --full-ISO)\";" >>
  compilation_date.ads
  echo "end Compilation_Date;" >>
   compilation_date.ads
```

This is a simple and valid approach but works only for things that are build-wide. The other extremely useful macros from the C world are __FILE__ and __LINE__. I'm afraid there is no way to simulate them using any external solutions, short of using the real preprocessor (why not?).

Yeah — why not use the cpp preprocessor?

Let's say that instead of implementing the .adb file we write something that will be used to generate it:

```
$ cat a.adbs
    with Ada.Text_IO;
    procedure A is
    begin
      Ada.Text_IO.Put_Line
        (__DATE__ & " " & __TIME__);
    end A;
$ cpp -P a.adbs > a.adb
$ gnatmake a
gcc -c a.adb
gnatbind -x a.ali
gnatlink a.ali
$ ./a
Sep  9 2008 22:53:23
```

Works for me. I can even write a rule in Makefile to make (pun intended) that automated. Not sure if GNAT project files can do it, though.

From: Per Sandberg <per.sandberg@bredband.net>
Date: Wed, 10 Sep 2008 06:57:39 +0000
Subject: Re: C Macros and their equivalent in Ada
Newsgroups: comp.lang.ada

If using GNAT the simple approach for those two macros could be found in the package:

GNAT.Source_Info

where you could find:

```
function File return String; -- Current file
function Line return Positive; -- Current
                                  Line
function Source_Location
  return String; -- Current file & Line
function Enclosing_Entity
  return String; -- Enclosing name.
```

From: Keith Thompson <kst-u@mib.org>
Date: Wed, 10 Sep 2008 12:22:06 -0700
Subject: Re: C Macros and their equivalent in Ada
Newsgroups: comp.lang.ada

The cpp preprocessor is designed to be used with C and C++.

For example, the apostrophe character in C, if it appears outside a string literal, is used only as a delimiter for a character constant. If you have a single apostrophe on a line because you're using an Ada qualified expression or attribute, or even in an Ada comment (the leading — will be treated as an operator symbol), then cpp is likely to complain or even terminate.

On implementation I just tried, it issued a warning message and continued processing, but that's not guaranteed.

Interfacing to C without dynamic memory

From: Maciej Sobczak <maciej@msobczak.com>
Date: Fri, 14 Nov 2008 05:58:03 −0800 (PST)
Subject: Interfacing to C without dynamic memory
Newsgroups: comp.lang.ada

[…] Consider a C (or C++) library that defines some type T with some functions operating on it. The type T can be a complex type, encapsulated in a struct or class. Assuming extern “C” interface, the library functions usually accept a pointer to T:

```
struct T
{
  // lots of interesting stuff
};
void createT(T * object);
void destroyT(T * object);
void foo(T * object, int something);
void bar(T * object, int something_else);
```

In C and C++ it is possible to use objects of type T without allocating them dynamically, which would be otherwise mandated by an alternative interface:

```
T * newT();
void deleteT(T * object);
```

How would you approach wrapping such a library for Ada while retaining the requirement that dynamic memory is not obligatory (ie. with the original interface above)?

My first ideas are:

1. Extract sizeof(T) at C level.

2. In Ada, create appropriately aligned:

```
type T is System.Storage_Array
    (1 .. Size_Of_T);
for T'Alignment use
    Appropriate_Alignment_Value;
```

3. Wrap imported C functions by passing to them the 'Address of T's instances.

This way, users will be able to use T as any other value type, without resorting to dynamic memory.

Does it make sense? Is there any better way?

From: Damien Carbonne <damien.carbonne@free.fr>
Date: Fri, 14 Nov 2008 21:35:33 +0100
Subject: Re: Interfacing to C without dynamic memory
Newsgroups: comp.lang.ada

[…] I never tried what you propose, but I think it should work. Of course, all fields of T should be initialized in C code (in createT, I guess).

In Ada, you could also wrap the array in a record. That way, you could initialize everything to 0:

```
type T is record
   Bytes :  System.Storage_Array (…);
end record;
pragma Convention (C, T);
```

+ other pragmas if necessary

…

Make T private, or even limited private: Ada users should not have direct access to T fields. Also, you shouldn't need to use 'Address. By default, when you use C convention/import, structures and arrays are passed by address.

Of course, as you don't have direct access to T fields, you need to provide all necessary functions to do that.

So you would have:

```
package XXX is
```

```
   type T is [limited] private;
   procedure Create (O : in out T);
   procedure Destroy (O : in out T);
   …
private
   type T is … -- as your proposal
   …
   pragma Import (C, Create,
                  "CreateT");
   pragma Import (C, Destroy,
                  "DestroyT");
   …
end XXX;
```

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Fri, 14 Nov 2008 19:12:33 −0600
Subject: Re: Interfacing to C without dynamic memory
Newsgroups: comp.lang.ada

> 3. Wrap imported C functions by passing to them the 'Address of T's instances.

You were fine up to here. But there is no reason at all to do this; you can use 'Access and a general access type with the appropriate convention; 'Access and an anonymous access type, or best of all, just plain parameters (which will be passed by reference for routines with the C convention).

An Ada 95 or newer program that uses System.Address outside of address clauses is broken, IMHO. There are many safer ways to do those things. (And avoiding access types is good, too.)

For Claw, most of the interfaces use locally declared records with the StdCall convention (and all of the components defined by Win32), and usually normal 'in' and 'in out' parameters. The only access parameters that we used were in functions that modified some of their arguments (and that was only because of Ada's broken rule against 'in out' parameters in functions).

From: Robert A Duff <duff@adacore.com>
Date: Fri, 14 Nov 2008 18:13:59 −0500
Subject: Re: Interfacing to C without dynamic memory
Newsgroups: comp.lang.ada

> My first ideas are:

1. Extract sizeof(T) at C level.
2. In Ada, create appropriately aligned:

```
type T is System.Storage_Array (1 ..
  Size_Of_T);
for T'Alignment use
  Appropriate_Alignment_Value;
```

Makes sense. You could write a C program that #include's the relevant .h file, and prints out an Ada package spec containing the above Ada code. You could run this as part of your build scripts.

From: Samuel Tardieu <sam@rfc1149.net>
Date: Sat, 15 Nov 2008 12:52:54 +0100
Subject: Re: Interfacing to C without dynamic memory
Newsgroups: comp.lang.ada

> […] You could write a C program that #include's the relevant .h file, and prints out an Ada package spec containing the above Ada code. […]

Doing it properly is quite painful when you cross-compile, you have to play dirty unportable tricks such as parsing the assembly file output by the compiler to extract the value. This is the way I originally did it in GLADE, it was then used to generate GNAT.Sockets.Constants, but as you know AdaCore recently switched to parsing the C compiler output, which was possible because the GCC C compiler (native or cross) is always available when building the runtime.

Note that you can also do it at run time, even if it costs a few more cycles to allocate the object, by taking advantage of C ability to resolve "sizeof" into a static value at compile time. This obliviates the cross-compilation issues. Here is an example (t.c, t_interface.ads and test.adb): (note that in real life, one would include the header defining "struct T" instead of exporting "T_size" in the same file)

```
/* t.c */
#include <stdlib.h>
struct T {
  char a;
  void *b;
  short c;
  char d;
};
const size_t T_size = sizeof(struct T);
```

```
-- t_interface.ads
with Interfaces.C;
package T_Interface is
   T_Size : constant Interfaces.C.size_t;
   pragma Import (C, T_Size, "T_size");
   type Opaque_T is new
      Interfaces.C.char_array
      (1 .. T_Size);
   for Opaque_T'Alignment use 8;
private
  pragma Linker_Options ("t.o");
end T_Interface;
```

```
-- test.adb
with Ada.Text_IO; use Ada.Text_IO;
with T_Interface; use T_Interface;
procedure Test is
   My_T : Opaque_T;
begin
   Put_Line ("Size of T in bits:" &
             My_T'Size'Img);
end Test;
```

Array of Strings

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Sat, 13 Sep 2008 16:32:54 +0200
Subject: Re: Array of Strings
Newsgroups: comp.lang.ada

[…]

> I'm having trouble making an array of type string. Can someone show me how to do this? I've tried type letters is array (Integer range <>) of String; but I get error "unconstrained element type in array declaration". What am I doing wrong?

The type String is unconstrained because you don't know the size of the strings at compile time. Therefore you cannot put Strings in an array. You can however create:

- an array of fixed-size strings; for this you need a subtype e.g.

```
subtype Constrained_String is
  String (1 .. 10);
type Array_Of_Constrained_Strings is
  array (Positive range <>) of
  Constrained_String;
```

- an array of access values to Strings e.g.

```
type String_Access is access String;
type Array_Of_String_Accesses is
   array  (Positive range <>) of
   String_Access;
```

(!this is the most error-prone method!)

- an array of Ada.Strings.Unbounded. Unbounded_Strings e.g.

```
type Array_Of_Unbouded_Strings is
   array (Positive range <>) of
   Ada.Strings.Unbounded.
   Unbounded_String;
```

- an array of Ada.Strings.Bounded. Bounded_Strings similar to the above but you also need to specify the maximum size

From: Adam Beneschan <adam@irvine.com>
Date: Mon, 15 Sep 2008 07:54:10 −0700 (PDT)
Subject: Re: Array of Strings
Newsgroups: comp.lang.ada

[…] One of the things that needs to be impressed upon newer Ada programmers is that Ada's String type is *not* a varying-length string. Some languages have varying-length strings built into them, like BASIC and Perl; you can just assign a 2-character string into a string variable, and later change it to an 80-character string, and later to an empty (zero-length) string, and so on, without worrying too much. (I believe there are many other languages that also have this sort of string type.) Ada's String type is not like this. A String variable has a

particular length and that length cannot be changed. (Maybe it was a mistake to name it “String” because of the potential for confusion.)

Ludovic already mentioned Bounded_String and Unbounded_String which were added in Ada 95 to fill that hole.

From: John McCormick <mccormick@cs.uni.edu>
Date: Wed, 24 Sep 2008 08:13:12 -0700 (PDT)
Subject: Re: Array of Strings
Newsgroups: comp.lang.ada

In a bit of a self serving suggestion, have a look at Chapter 4 of my Freshman level textbook “Ada Plus Data Structures: An Object-Oriented Approach”, Dale and McCormick, Jones and Bartlett, 2007. This chapter is devoted to the use and implementation of strings: Fixed-Length, Bounded-Length, and Unbounded-Length. You can check it out for free on Google Books.

Changes since Ada 83

From: Stephen Horne <sh006d3592@blueyonder.co.uk>
Date: Thu, 25 Sep 2008 16:57:17 +0100
Subject: changes “in a nutshell” since Ada 83
Newsgroups: comp.lang.ada

I last used Ada about 10 years ago, and only ever used Ada 83. Is there a quick-and-simple summary of what's changed in Ada 95 and since that I can download?

What I really want is something short-and-sweet and focused on the main changes — what changed, quick rationale, quick example, move on.

[…]

I have found some books available online, but they all assume the reader has never used Ada at all, and also that the reader has never programmed before — not that that's an unusual assumption in programming language books for some reason.

From: Stephen Horne <sh006d3592@blueyonder.co.uk>
Date: Thu, 25 Sep 2008 17:21:30 +0100
Subject: Re: changes “in a nutshell” since Ada 83
Newsgroups: comp.lang.ada

> Maybe the Rationale can help?

Ada 95:
http://www.adaic.org/standards/95rat/RAThtml/rat95-contents.html

Ada 2005:
http://www.adaic.org/standards/05rat/html/Rat-TTL.html

Pretty close to perfect — thanks.

Finalization and ATC

From: Florian Weimer <fw@deneb.enyo.de>
Date: Sat, 01 Nov 2008 12:13:01 +0100
Subject: Storage management
Newsgroups: comp.lang.ada

What is the current state of the art with regard to programmer support for storage management?

With GNAT, Ada.Finalization adds tons of run-time calls to deal with abort deferral (even with pragma Restrictions (No_Abort_Statements)), puts the object on some sort of list, and does some secondary stack allocations which I don't understand. Clearly, this is not supposed to be used in performance-critical code, so I doubt it can be used for a generic smart pointer implementation. (Object allocation in inner loops is generally a bad idea, but this overhead is also incurred when copying smart pointers around.)

Is there some other approach? Can limited types be used to enforce linearity (in the sense of linear types, cf. http://home.pipeline.com/~hbaker1/Use1Var.html and http://iml.univ-mrs.fr/~girard/linear.pdf)?

If it's about pure storage management, GC support would be an option, too, but as far as I know, you need one of the managed code implementations for that, or somewhat unsafe libraries (in the sense that you need to specify which objects are leaf objects, and only store access values in locations where they are visible to the collector).

From: Robert A Duff <duff@adacore.com>
Date: Sat, 01 Nov 2008 18:28:04 −0400
Subject: Re: Storage management
Newsgroups: comp.lang.ada

AdaCore is actively working on making finalization much more efficient (e.g. avoiding all those finalization lists, when possible).

> Is there some other approach?

Yes, you can do better with limited types (derive from Limited_Controlled instead of Controlled).

There is some intention to support proper GC at AdaCore, but it's going to be rather far in the future — not a whole lot of (paying) customer demand for that. Meanwhile, you can try the Boehm “conservative” GC. My experience is that it works pretty well, so long as you don't use up most of your address space with allocated objects. So on a 64-bit machine, that should work pretty well. Or on a 32-bit machine, if you don't use too much real memory.

The ARG is also working on some interesting “region based” schemes.

From: “Randy Brukardt” <randy@rrsoftware.com>
Date: Thu, 6 Nov 2008 19:14:40 −0600
Subject: Re: Storage management
Newsgroups: comp.lang.ada

> This sounds interesting. Unfortunately, the language-mandated overhead (primarily abort deferral) is difficult to get rid of. I hope that there will be a configuration pragma which eliminates it, even if it means using a self-compiled run-time.

Actually, that's the easiest to get rid of (or minimize enough to make it irrelevant most of the time). The Janus/Ada implementation of controlled types is roughly the same as the GNAT one. We take two steps to reduce the overhead of abort-deferral:

(1) If the program has no tasks, there is no task supervisor and the abort deferral routine does nothing (it gets called, but the overhead is just an indirect call and a return instruction). We optimized programs not containing tasks so that we have less of a disadvantage in benchmarks against other non-tasking languages (like C and C++). (It's not as common for real Ada programs to not contain tasks, but it helps any that don't as well.)

(2) We made abort deferral as cheap as possible. It is just a counter in the TCB of a task, and we change it directly in the supervisor interface code (this is a big advantage of not use OS threads for task mapping). It takes 4 machine instructions (this is assembler code). Re-enabling aborts is slightly more expensive, as we have to check if someone did abort the task while it was abort deferred. But that is the rare case, and it add only two machine instructions.

Actually adding or removing an object from the finalization chain isn't that expensive, either. It takes about 10 machine instructions.

The biggest expense with Janus/Ada is putting the needed exception handler around a Finalize call (we have to turn all exceptions into Program_Error). That's probably cheaper with GNAT.

I find I have more sympathy with those that worry about the 16-byte per object space overhead (that can be significant for smart pointers, for instance, if there are a lot of them in the program). The time will matter only in the most critical of applications. (And, based on another thread here, moving a few bytes around in some random location will change the performance of your program +/− 50% anyway — that effect will completely swamp any finalization overhead.) I doubt anyone with truly critical performance needs is going to be using smart pointers or containers or anything else that adds overhead.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Fri, 07 Nov 2008 11:54:05 +0200
Subject: Re: Storage management
Newsgroups: comp.lang.ada

> I have a general question. Does anybody use abort and asynchronous transfer of control?

I do, to set a limit on the running time of a possibly lengthy procedure Analyse:

```
select
   delay Opt.Max_Analysis_Time;
   Output.Error ("Maximum analysis
                  time exceeded.");
then abort
   Analyse;
end select;
```

> After all, there is no chance to have them reasonably working anyway.

It seems to work for me (GNAT 3.15p), except that I had to add a dummy "delay 0.1" at the end of the main subprogram. Otherwise the program would sometimes hang indefinitely in the termination phase. This program has no tasks, so the abort and ATC are the only form of concurrency.

> How about to remove that stuff altogether and make a far more important finalization right and fast?

Isn't pragma Restrictions (No_Select_Statements) enough? Or is it too strong, and a new restriction specifically for ATC would be better?

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Fri, 07 Nov 2008 14:12:30 +0200
Subject: Re: Storage management
Newsgroups: comp.lang.ada

> That does not look like a good example. In such cases there would be some GUI with progress indication stuff, etc. I mean that most likely Analyse would periodically call something in order to indicate its state; store the results etc. These would be natural candidates to abort it "cooperatively," through an exception propagation.

This example is a batch program — no GUI, no interaction. The most unpredictable part of the execution time is spent waiting for a child process to respond, using blocking I/O to read a pipe that carries the standard output channel of the child process. The child process can get stuck (take a very long time) at any point, so it is not enough to make Analyse check the elapsed time after every pipe-read, for example.

If this example had a GUI, it would not need a programmed delay-then-abort time-out; the user would get bored and would click something to abort the child process, which would make the Analyse procedure terminate, too.

> I honestly believe that the only case that may justify abort/ATC is cancellation of an outstanding blocking I/O.

Which is the case in my example.

I think that abort/ATC is also useful in hard real-time systems as a guard against a task overrunning its deadlines (although execution-time budgeting is an alternative, perhaps better). It is difficult and error-prone to embed overrun-checking code in the task itself, and it will complicate the code — poor "separation of concerns".

> But exactly this case is not guaranteed to work, or rather is guaranteed not to work…

Aborting blocking I/O is "guaranteed not to work"? Can you explain why? Is this something that has been discussed before on c.l.a.?

>> Isn't pragma Restrictions (No_Select_Statements) enough? Or is it too strong, and a new restriction specifically for ATC would be better?

> I would prefer pragma Cancelable put on a task. If a task is not cancelable then abort would raise Tasking_Error, and an ATC in the task body would be a compile-time error.

I will leave it to the language experts to comment on that suggestion. I assume it would have to forbid ATCs in subprograms called from the task, too, which would require some form of subprogram-level contract that the subprogram body executes no ATCs.

What about ctrl-C, that is, process abort from the operating system? I don't think that users would be happy if Ada applications could not be aborted with ctrl-C. Do abort-deferred operations now defer ctrl-C, too?

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Fri, 7 Nov 2008 14:22:36 +0100
Subject: Re: Storage management
Newsgroups: comp.lang.ada

> Aborting blocking I/O is "guaranteed not to work"? Can you explain why? Is this something that has been discussed before on c.l.a.?

I don't know. But I discussed that with AdaCore people. Since the following does not work with GNAT Pro 6.2 (Windows wavefront):

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Test_ATC is
begin
   Put_Line ("Type");
   select
      delay 2.0;
      Put_Line ("Timed out");
   then abort
      Put_Line ("You typed " & Get_Line);
   end select;
end Test_ATC;
```

Actually I was almost certain that this were *not* required to work. I only wished to know if they considered it as nice to have working or not.

There is no way Ada could abort I/O if the OS does not allow this. Second to Get_Line, or likely the first wanted case is canceling blocking socket read. I would give 98% that it never will work with ATC.

> […] I assume it would have to forbid ATCs in subprograms called from the task, too, which would require some form of subprogram-level contract that the subprogram body executes no ATCs.

Or to constrain thir use like with the selective accept.

> What about ctrl-C, that is, process abort from the operating system? I don't think that users would be happy if Ada applications could not be aborted with ctrl-C. Do abort-deferred operations now defer ctrl-C, too?

The meaning of ctrl-C depends on the OS. Ada standard cannot mandate that pressing ctrl-C is equivalent to aborting the environment task in the sense of "abort" statement. Then usually OS have efficient means to kill a process preemptively bypassing any "abort deferred" stuff.

From: "Dmitry A. Kazakov" <mailbox@dmitry-kazakov.de>
Date: Mon, 10 Nov 2008 17:08:53 +0100
Subject: Re: Storage management
Newsgroups: comp.lang.ada

>> I don't know. But I discussed that with AdaCore people.

> How come? What did they say?

I think they will not object me quoting the conversation:

Q. Is it correct to conclude that Get_Line is not abortable this way?

A. Right, that's a correct understanding.

In general it is unsafe or not possible to interrupt system calls, so if you need to interrupt I/O, you should probably use something like Get_Immediate.

I think they are absolutely right here. It may work (under SuSe), or not (under Windows). To me this is a perfect reason to remove ATC stuff from the language. It causes a sufficient distributed overhead, is unreliable with OO programming and non-portable in virtually single case where it could come handy.

Stack usage at run time

From: Andy Vontovel <andi.vontobel@gmx.ch>
Date: Mon, 3 Nov 2008 23:55:26 −0800 (PST)
Subject: Stackusage at runtime
Newsgroups: comp.lang.ada

[…] Is it possible to figure out how much of the stack is used at a specific moment?

-> Ada 95 Cross-compiler to PPC

(It is a little bit hard to optimize the stack size …)

From: Andy Vontovel <andi.vontobel@gmx.ch>
Date: Tue, 4 Nov 2008 04:56:22 −0800 (PST)
Subject: Re: Stackusage at runtime
Newsgroups: comp.lang.ada

> I'd guess that your debugger can stop the program at that specific moment and tell you about the stack.

The debugger is not really usable … is there a possibility to read the value in the code and to print out?

From: Stuart
Date: Tue, 4 Nov 2008 15:40:46 -0000
Subject: Re: Stackusage at runtime
Newsgroups: comp.lang.ada

Compare the value of the stack pointer to its base value!

For a PowerPC it is very likely that the code will be conforming to the Embedded Application Binary Interface (EABI), so the stack pointer will be register r1 (aka gpr1). Reading this will require a bit of machine code — either use package machine code or call out to an assembler routine depending on how 'pure' you want to be.

Making this a subprogram would allow you to quickly find the stack pointer value anywhere in your program. Depending on what you actually need you might need to make adjustments for the stack frame of the subprogram returning the value (but I would suspect this is quite small and the discrepancy introduced to be of minimal interest if you are optimizing the programme).

The base value of the stack is often defined by a symbol (you will need to check your compiler's documentation) though take care as to whether you need the address represented by the symbol or the value at that address (in my experience it is usually the former).

As a pedagogical point — the Ada language does not really define what the stack (if any) is used for, so your results may not be universally applicable (but I suspect that is not a significant concern).

From: Per Sandberg <per.sandberg@bredband.net>
Date: Tue, 04 Nov 2008 21:07:34 +0100
Subject: Re: Stackusage at runtime
Newsgroups: comp.lang.ada

If you are running GNAT it is quite straight forward and described in detail in the top secret document (The GNAT users guide chapter 22. Stack Related Facilities)

In short

1 Compile the code with:

```
gcc -c -fstack-check *.adb
```

2 Bind your executable with

```
gnatbind -u0 file
```

3 Set environment variable

```
GNAT_STACK_LIMIT
```

4 Run your program and terminate clean.

5 Done.

I expect that all targets behave more ore less the same.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Tue, 04 Nov 2008 22:44:20 +0200
Subject: Re: Stackusage at runtime
Newsgroups: comp.lang.ada

Oh, you want to know how much stack-space a task needs overall (worst case), not how much is in use at a particular point in the execution? If so, how about GNATstack:

http://www.adacore.com/home/gnatpro/add-on_technologies/stack_analysis/

Or AbsInt's StackAnalyzer:

http://www.absint.com/stackanalyzer/

From: Andy Vontovel <andi.vontobel@gmx.ch>
Date: Tue, 4 Nov 2008 23:33:32 −0800 (PST)
Subject: Re: Stackusage at runtime
Newsgroups: comp.lang.ada

> If you are running GNAT it is quite straight forward and described in detail in the top secret document (The GNAT users guide chapter 22. Stack Related Facilities)

[…]

We are using Apex … Probably we think about AdaCore's static analysis tool …

I thought that Ada itself has a possibility to report the memory-usage at a specific time.

From: Per Sandberg <per.sandberg@bredband.net>
Date: Wed, 05 Nov 2008 17:50:32 +0100
Subject: Re: Stackusage at runtime
Newsgroups: comp.lang.ada

[…] Concerning Apex, once upon a time in the dark ages (80:ties & 90:ties) we were using Apex as well for embedded systems. What we did at that point in time was something along these lines:

- On program start, fill the memory allocated for the stacks with 16#CACA#.
- After some time of execution, query the operation system /Runtimes of the following, Stack start & Stack Size very very target dependent.
- Iterate from the free side of the allocated memory for each stack until no more "CACA" is found.
- That's how we did it then.

And that is actually the same basic method used in the more modern GCC/GNAT system to obtain the same result.

But if it's possible to do static analysis with GNATstack that would be much better.

Concerning Ada and stacks, as far as I know there is nothing defined in the language.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Wed, 05 Nov 2008 20:50:32 +0200
Subject: Re: Stackusage at runtime
Newsgroups: comp.lang.ada

> We are using Apex … Probably we think about AdaCore's static analysis tool …

As I understand it, AdaCore's GNATstack is coupled with GNAT, and uses compile-time information from GNAT; it may not be possible to use it with the Apex compiler. Of course, code generated by GNAT may not use the same amount of stack space as code generated by Apex.

The AbsInt StackAnalyzer works on the machine code alone, I believe, so it should in principle work with any compiler. However, I don't know if it handles the "secondary stack" used by some Ada compilers. Apex is not listed as a supported compiler on AbsInt's web, but it may well work.

On record serialization

From: MOE37x3
Date: Wed, Sep 17 2008 15:43
Subject: Understanding how Ada serializes a record
URL: http://stackoverflow.com/questions/84677/understanding-how-ada-serializes-a-record/

I would like to be able to predict what will be in the resulting binary when I call Write in Ada to serialize a record. Do you know where I can look this up?

I have some legacy Ada software that produces a binary file by Write-ing a record, and I need to debug a C++ program that is supposed to write a compatible binary file. So, I would like to understand what rules Ada follows when it serializes a record, so that I can make sure that the C++ code will produce a functionally equivalent record.

From: Mike Dimmick
Date: Wed, Sep 17 2008 16:06
Subject: Understanding how Ada serializes a record
URL: http://stackoverflow.com/questions/84677/understanding-how-ada-serializes-a-record/

The Ada 95 Language Reference Manual says (section 13.13.2): "For elementary types, the representation in terms of stream elements is implementation defined. For composite types, the Write or Read attribute for each component is called in a canonical order. The canonical order of components is last dimension varying fastest for an array, and positional aggregate order for a record."

From: Dan
Date: Wed, Sep 17 2008 16:20
Subject: Understanding how Ada serializes a record
URL: http://stackoverflow.com/questions/84677/understanding-how-ada-serializes-a-record/

Basically, the compiler will reorder the components of your record types, unless you use the pragma PACK or the pragma PRESERVE_LAYOUT commands with your record types. Also, the compiler will pad objects to maintain the alignment of record components. Components follow:

- Integer: 8, 16, or 32 bit twos-complement signed numbers
- Float: 32-bit IEEE format
- Long_Float: 64-bit IEEE format
- Fixed-Point: 8, 16, or 32 bit; however, the range and delta specified can affect being 16 or 32
- Enumerations: Integer, usually first element is represented by 0
- Booleans: Enumeration object, 8 bits long, The LSB stores the value: 0 = false, 1 = true
- Characters: Enumeration object, 8 bits long, unsigned 0 through 127
- Access Types: 32 bits, 32-bit value of 0 represents NULL
- Arrays: stored contiguously in row-major order, size depends on base type. The array is padded to ensure all elements have the proper alignment for their types.

From: Greg Hewgill
Date: Wed, Sep 17 2008 22:19
Subject: Understanding how Ada serializes a record
URL: http://stackoverflow.com/questions/84677/understanding-how-ada-serializes-a-record/

As mentioned by others, without additional instruction the compiler will make its own decisions about record layout. The best approach would be to change the original code to write the record using a specific layout. In particular, the record representation clause allows the Ada programmer to specify exactly the physical layout for a record. In fact, you should check to see whether the original code has one of these for the type in question. If it does, then this would answer your question precisely.

From: Simon Wright <simon.j.wright.myopenid.com>
Date: Tue, Nov 25 2008 23:29
Subject: Understanding how Ada serializes a record
URL: http://stackoverflow.com/questions/84677/understanding-how-ada-serializes-a-record/

The format of the serialised output of 'Write has nothing to do with representation clauses.

By default, the compiler will output record components without alignment padding in the order in which they're written in the record declaration, using a translation scheme that isn't defined by the standard (so you may not get interoperability between compilers). GNAT (the GCC Ada compiler) outputs each component in a whole number of bytes.

If you want to stream values of a type using some different format, you can override 'Write for the type. As an unusual example, you could stream to XML.

Does Ada have a preprocessor?

From: ceretullis
Date: Wed, 12 Nov 2008 13:13
Subject: Does Ada have a preprocessor?
URL: http://stackoverflow.com/questions/283893/does-ada-have-a-preprocessor/

To support multiple platforms in C/C++, one would use the preprocessor to enable conditional compiles. E.g.,

```
#ifdef _WIN32
        #include <windows.h>
#endif
```

How can you do this in Ada? Does Ada have a preprocessor?

From: David Allan Finch
Date: Wed, 12 Nov 2008 13:27
Subject: Does Ada have a preprocessor?
URL: http://stackoverflow.com/questions/283893/does-ada-have-a-preprocessor/

No but the CPP preprocessor or m4 can be called on any file on the command line or using a building tool like make or ant. I suggest calling your .ada file something else. I have done this for some time on java files. I call the java file .m4 and use a make rule to create the .java and then build it in the normal way.

From: Marc C
Date: Tue, 18 Nov 2008 21:27
Subject: Does Ada have a preprocessor?
URL: http://stackoverflow.com/questions/283893/does-ada-have-a-preprocessor/

AdaCore provides the gnatprep preprocessor, which is specialized for Ada. They state that gnatprep "does not depend on any special GNAT features", so it sounds as though it should work with non-GNAT Ada compilers. Their User Guide also provides some conditional compilation advice.

I have been on a project where m4 was used as well, with the Ada spec and body files suffixed as ".m4s" and ".m4b", respectively. My preference is really to avoid preprocessing altogether, and just use specialized bodies, setting up CM and the build process to manage them.

From: lbrandy
Date: Thu, 20 Nov 2008 13:57
Subject: Does Ada have a preprocessor?
URL: http://stackoverflow.com/questions/283893/does-ada-have-a-preprocessor/

The answer to your question is no, Ada does not have a pre-processor that is built into the language. That means each compiler may or may not have one and there is not "uniform" syntax for pre-processing and things like conditional compilation. This was intentional: it's considered "harmful" to the Ada ethos. There are almost always ways around a lack of a preprocessor but often times the solution can be a little cumbersome. For example, you can declare the platform specific functions as 'separate' and then use build-tools to compile the correct one (either a project system, using pragma body replacement, or a very simple directory system… put all the windows files in /windows/ and all the linux files in /linux/ and include the appropriate directory for the platform). All that being said, GNAT realized that sometimes you need a preprocessor and has created gnatprep. It should work regardless of the compiler (but you will need to insert it into your build process). Similarly, for simple things (like conditional compilation) you can probably just use the c pre-processor or even roll your own very simple one.

From: ted.dennison.myopenid.com
Date: Thu, 20 Nov 2008 14:23
Subject: Does Ada have a preprocessor?
URL: http://stackoverflow.com/questions/283893/does-ada-have-a-preprocessor/

No, it does not.

If you really want one, there are ways to get one (Use C's, use a stand-alone one, etc.) However I'd argue against it. It was a purposeful design decision to not have one. The whole idea of a preprocessor is very un-Ada. Most of what C's preprocessor is used for can be accomplished in Ada in other more reliable ways. The only major exception is in making minor changes to a source file for cross-platform support. Given how much this gets abused in a typical cross-platform C program, I'm still happy there's no support for it in Ada. Very few C/C++ developers can control themselves enough to keep the changes "minor". The result may work, but is often nearly impossible for a human to read.

The typical Ada way to accomplish this would be to put the different code in different files and use your build system to somehow choose between them at compile time. Make is plenty powerful enough to help you do this.

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

January 21-23 — 36th ACM **SIGPLAN-SIGACT Symposium on Principles of Programming Languages** (POPL'2009), Savannah, Georgia, USA. Topics include: fundamental principles and important innovations in the design, definition, analysis, transformation, implementation and verification of programming languages, programming systems, and programming abstractions.

January 24 — 2009 International **Workshop on Foundations of Object-Oriented Languages** (FOOL'2009), Savannah, Georgia, USA. Following POPL'2009. Topics include: language semantics, type systems, program analysis and verification, concurrent and distributed languages, language-based security issues, etc.

☺ February 04-06 — **International Symposium on Engineering Secure Software and Systems** (ESSoS'2009), Leuven, Belgium. Topics include: security architecture and design for software and systems; systematic support for security best practices; programming paradigms, models and DLS's for security; program rewriting techniques; processes for the development of secure software and systems; etc. Includes full-day tutorial on "Security by Construction" by Rod Chapman, Praxis High Integrity Systems, Bath. Deadline for early registration: January 6, 2009.

♦ Feb 07-08 — **Ada at the Free and Open-Source Software Developers' European Meeting** (FOSDEM'2009), Brussels, Belgium. FOSDEM 2009 is a two-day event (Sat-Sun 07-08 February). This years' edition includes again an Ada track, organized by Ada-Belgium, and will run over both days of the event.

February 16-19 — 7th **International Conference on Integrated Formal Methods** (IFM'2009), Düsseldorf, Germany. Deadline for early registration: January 8, 2009.

February 17-19 — 22nd IEEE-CS **Conference on Software Engineering Education and Training** (CSEET'2009), Hyderabad, India. Theme: "Scalability in Software Engineering Education and Training". Topics include: Curriculum and teaching materials, Learning environments, Software engineering professionalism, Case studies of educational or training practices, Industry-academia collaboration models, etc.

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

March 08-12 — 24th ACM **Symposium on Applied Computing** (SAC'2009), Honolulu, Hawaii, USA.

- ☺ Mar 08-12 — **Track on Software Engineering** (SE'2009). Topics include: Component-Based Development and Reuse; Safety and Security Dependability and Reliability; Fault Tolerance and Availability; Design Patterns; Standards; Maintenance and Reverse Engineering; Verification, Validation, and Analysis; Formal Methods and Theories; Empirical Studies and Industrial Best Practices; Applications and Tools; Distributed, Embedded, Real-Time, High Performance, and Highly Dependable Systems; etc.
- ☺ Mar 08-12 — **Track on Object-Oriented Programming Languages and Systems** (OOPS'2009). 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.

☺ Mar 08-12 **Track on Real-Time Systems** (RTS'2009). Topics include: 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.

☺ Mar 08-12 **Track on Programming Languages** (PL'2009). Topics include: Compiling Techniques, Formal Semantics and Syntax, Language Design and Implementation, Model-Driven Development and Model Transformation, New Programming Language Ideas and Concepts, Practical Experiences with Programming Languages, Program Analysis and Verification, Program Generation and Transformation, Programming Languages from All Paradigms, etc.

Mar 08-12 **Track on Software Verification and Testing** (SVT'2009). Topics include: 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.

March 16-19 3rd **International Conference on Complex, Intelligent and Software Intensive Systems** (CISIS'2009), Fukuoka, Japan.

☺ Mar 16-19 **International Workshop on Multi-Core Computing Systems** (MuCoCoS'2009). Topics include: multi-core embedded systems; programming languages and models; applications for multi-core systems; performance modeling and evaluation of multi-core systems; design space exploration; tool-support for multi-core systems; compilers, runtime and operating systems; etc.

Mar 16-19 1st **Workshop on Coordination in Complex Software Intensive Systems** (COCOSS'2009). Topics include: Distributed problem solving, Programming abstractions and languages, Case studies, etc.

☺ March 17-20 12th IEEE **International Symposium on Object/component/service-oriented Real-time distributed Computing** (ISORC'2009), Tokyo, Japan. Topics include: Programming and system engineering (ORC paradigms, languages, RT Corba, UML, model-driven development of high integrity applications, specification, design, verification, validation, testing, maintenance, system of systems, etc.); System software (real-time kernels, middleware support for ORC, extensibility, synchronization, scheduling, fault tolerance, security, etc.); Applications (embedded systems (automotive, avionics, consumer electronics, etc), real-time object-oriented simulations, etc.); System evaluation (timeliness, worst-case execution time, dependability, fault detection and recovery time, etc.); ...

March 22-29 12th **European Joint Conferences on Theory and Practice of Software** (ETAPS'2009), York, UK.

March 22 8th **International Workshop on Compiler Optimization Meets Compiler Verification** (COCV'2009). Topics include: optimizing and verifying compilation, and related fields such as translation validation, certifying and credible compilation, programming language design and programming language semantics, etc.

March 22-29 18th **European Symposium on Programming** (ESOP'2009). Topics include: issues in the specification, design, analysis, and implementation of programming languages and systems, such as Programming paradigms and styles (object-oriented programming, real-time programming languages, etc), Methods and tools to write, reason about, and specify languages and programs (module systems, programming techniques, type systems, program verification, static analysis, language-based security, etc), Methods and tools for implementation, Concurrency and distribution (parallel programming, distributed languages, etc).

March 22-29 18th **International Conference on Compiler Construction** (CC'2009). Topics include: research on compilers in the broadest possible sense, including run-time techniques, programming tools, domain-specific languages, novel language constructs and so on.

March 23-29 12th **International Conference on Fundamental Approaches to Software Engineering** (FASE'2009). Topics include: novel techniques and the way in which they contribute to making Software Engineering a more mature and sound discipline.

Fundamental approaches, including: Software Engineering as an engineering discipline; Specification, design, and implementation of particular classes of systems (collaborative, embedded, distributed, ...); Software quality (validation and verification of software using theorem proving, model-checking, testing, analysis, metrics, ...); Software evolution (refactoring, reverse and re-engineering, configuration management, ...); etc.

March 28 — 2nd **Workshop on Verification and Analysis of Multi-threaded Java-like Programs** (VAMP'2009). Topics include: verification and analysis techniques for Multi-threaded Java-like languages, including automatic verification and static analysis techniques; type-based verification; specification techniques; race condition detection, deadlock detection, etc.; static analysis for bug discovery; etc.

March 28-29 — 6th **International Workshop on Formal Engineering approaches to Software Components and Architectures** (FESCA'2009). Topics include: software quality attributes such as reliability, performance, or security; interface compliance; approaches for correctness by construction; static and dynamic analysis; runtime management of applications; etc.

March 28-29 — 8th **Workshop on Language Descriptions, Tools and Applications** (LDTA'2009). Topics include: applications of and tools for meta programming in a broad sense, such as Program analysis, transformation, generation and verification; Reverse engineering and reengineering; Refactoring and other source-to-source transformations; Language definition and language prototyping; Debugging, profiling and testing; IDE construction; Compiler construction; etc. Deadline for early registration: February 13, 2009.

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

March 24-27 — 15th **French-speaking Conference on Object-Oriented Languages and Models** (LMO'2009), Nancy, France.

March 24-27 — 13th **European Conference on Software Maintenance and Reengineering** (CSMR'2009), Kaiserslautern, Germany. Topics include: Experience reports on maintenance and reengineering of large-scale software systems; Empirical studies in software reengineering, maintenance, and evolution; Education-related issues to evolution, maintenance and reengineering; etc.

☺ Mar 30- Apr 03 — 4th **European Conference on Computer Systems** (EuroSys'2009), Nuremberg, Germany. Topics include: All areas of operating systems and distributed systems; Systems aspects of: Dependable computing, Distributed computing, Parallel and concurrent computing, Programming-language support, Real-time and embedded computing, Security, ...; Experience with existing systems; Reproduction or refutation of previous results; Negative results; Early ideas. Deadline for submissions: January 5, 2009 (doctoral workshop papers), January 19, 2009 (other workshop papers).

April 01-04 — 2nd IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2009), Denver, Colorado. Topics include: Verification & Validation, Quality Assurance, Empirical studies, Embedded and real-time software, Concurrent software, etc.

April 06-08 — 2nd **International Conference on Trusted Computing** (Trust'2009), Oxford, UK. Topics include: implementation technologies for trusted platforms; implementations of trusted computing; verification of trusted computing architectures; etc.

Aprl 13-16 — 16th Annual IEEE **International Conference and Workshops on the Engineering of Computer Based Systems** (ECBS'2009), San Francisco, California, USA. Topics include: Component-Based System Design; Design Evolution; Distributed Systems Design; ECBS Infrastructure (Tools, Environments); Education & Training; Embedded Real-Time Software Systems; Formal Methods; Integration Engineering; Model-Based System Development; Modeling and Analysis of Complex

Systems; Open Systems; Reengineering & Reuse; Reliability, Safety, Dependability, Security; Standards; Verification & Validation; etc.

April 20-23 21st **Annual Systems and Software Technology Conference** (SSTC'2009), Salt Lake City, Utah, USA.

☺ May 16-24 31st **International Conference on Software Engineering** (ICSE'2009), Vancouver, Canada. Topics include: Specification and Verification; Software Architecture and Design; Patterns and Frameworks; Reverse Engineering, Refactoring, and Evolution; Tools and Environments; Empirical Software Engineering; Development Paradigms and Software Processes; Component-based Software Engineering; Model Driven Engineering; Distributed Systems and Middleware; Embedded System; Open Standards and Certification; Software Economics; Dependability (safety, security, reliability); Case Studies and Experience Reports; etc. Co-located events: 7th Workshop on Software Quality (WoSQ'2009), Workshop on Modeling in Software Engineering (MISE'2009), 2nd International Workshop on Multicore Software Engineering (IWMSE'2009), 2nd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS'2009), 5th International Workshop on Software Engineering for Secure Systems (SESS'2009), 2nd International Workshop on Software Engineering for Computational Science and Engineering (SECSE'2009), etc.

May 25-27 9th **International Conference on Computational Science** (ICCS'2009), Baton Rouge, Louisiana, USA. Theme: "Compute, Discover, Innovate". Deadline for early registration: March 15, 2009.

- ☺ May 25 6th **International Workshop on aPpplications of declArative and object-oriented Parallel Programming** (PAPP'2009). Topics include: high-level parallel language design, implementation and optimisation; modular, object-oriented, functional, logic, constraint programming for parallel, distributed and grid computing systems; industrial uses of a high-level parallel language; etc.
- ☺ May 25 **Workshop on Using Emerging Parallel Architectures for Computational Science**. Topics include: Languages, models, tools, and compilation techniques for emerging architectures; etc.

☺ May 25-29 23rd IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2009), Rome, Italy. Topics include: Parallel and distributed algorithms; Applications of parallel and distributed computing; Parallel and distributed software, including parallel programming languages and compilers, runtime systems, fault tolerance, middleware, libraries, scalability, programming environments and tools, etc.

☺ May 26-29 **DAta Systems In Aerospace** (DASIA'2009), Istanbul, Turkey.

June 03-06 5th **International Conference on Open Source Systems** (OSS'2009), Skövde, Sweden. Topics include: Software engineering perspectives (F/OSS development environments; Testing, assuring and certifying F/OSS quality and security; F/OSS usability, scalability, maintainability and other quality issues; F/OSS and standards, ...); Emerging perspectives (Licensing, IPR and other legal issues in F/OSS; F/OSS and innovation; ...); Studies of F/OSS deployment (Case studies of F/OSS deployment, migration models, success and failure; F/OSS in vertical domains and the 'secondary' software sector, e.g., automotive, telecommunications, medical devices; F/OSS applications catalog; ...); etc.

♦ June 08-12 14th **International Conference on Reliable Software Technologies – Ada-Europe 2009**, Brest, France. Sponsored by Ada-Europe, in cooperation with ACM SIGAda. Deadline for submissions: January 12, 2009 (industrial presentations).

June 09-12 4th IFIP **Conference on Distributed Computing Techniques** (DisCoTec'2009), Lisbon, Portugal.

- June 09-11 11th **International Conference on Languages, Models, and Architectures for Concurrent and Distributed Software** (Coordination'2009). Topics include: Distributed and Concurrent Programming Models (multicore programming, data parallel programming, event-driven programming, ...); Distributed Software Management (component and module systems for distributed software, configuration and deployment architectures, ...); Case Studies (application of novel distributed and concurrent techniques); etc. Deadline for submissions: January 28, 2009 (abstracts), February 1, 2009 (papers).
- June 09-11 IFIP **International Conference on Formal Techniques for Distributed Systems** (FMOODS/FORTE'2009). Formed joinlty from the 11th Formal Methods for Open

Object-Based Distributed Systems (FMOODS) and the 29th Formal Techniques for Networked and Distributed Systems (FORTE). Topics include: Languages and Semantic Foundations (new modeling and language concepts for distribution and concurrency, semantics for different types of languages, including programming languages, modeling languages, and domain specific languages; real-time aspects; ...); Formal Methods and Techniques (design, specification, analysis, verification, validation and testing of various types of distributed systems); Practical Experience with Formal Methods (industrial applications, case studies and software tools for applying formal methods and description techniques to the development and analysis of real distributed systems); etc. Deadline for submissions: January 28, 2009 (abstracts), February 1, 2009 (papers).

June 09-11 9th IFIP **International Conference on Distributed Applications and Interoperable Systems** (DAIS'2009). Topics include: Innovative distributed applications; Models and concepts supporting distributed applications; Middleware supporting distributed applications; Software engineering of distributed applications; etc. Deadline for submissions: January 28, 2009 (abstracts), February 1, 2009 (papers).

June 15-21 ACM SIGPLAN **Conference on Programming Language Design and Implementation** (PLDI'2009), Dublin, Ireland. Topics include: the design, development, implementation, evaluation, and use of programming languages; including: Extracting parallelism from programs, Exploiting explicit parallelism in programs, Memory management, Language constructs for parallelism, Program analyses, Type systems and program logics, Debugging techniques and tools, Language designs and extensions, Checking or improving the safety, security, or correctness of programs, etc.

June 16-18 **Code Generation 2009**, Cambridge, UK. Topics include: Tool and technology development and adoption; Code Generation and Model Transformation tools and approaches; Defining and implementing modelling languages; Language evolution and modularization; etc. Deadline for submissions: January 16, 2009 (speaker proposals).

June 22-26 29th **International Conference on Distributed Computing Systems** (ICDCS'2009), Montreal, Canada. Topics include: findings in any aspects of distributed and parallel computing, such as Distributed Middleware, Reliability and Dependability, Security, etc.

June 23-26 5th **European Conference on Model Driven Architecture Foundations and Applications** (ECMDA-FA'2009), Enschede, the Netherlands. Topics include: Metamodeling foundations and tools; Model Transformation and Code Geneation; MDA for Complex Systems and Systems of Systems; MDA for Embedded Systems and Real-Time Systems; MDA for High-Integrity Systems, Safety-Critical, and Security-Critical Systems; MDA in the Automotive, Aerospace, Telecommunications, Electronics Industries; Comparative Studies of MDA Methods and Tools; MDA for Legacy Systems; etc. Deadline for submissions: January 14, 2009 (workshops, tutorials); January 30, 2009 (abstracts); February 6, 2009 (papers); April 1, 2009 (ECMDA'2010 hosting proposals); April 6th, 2009 (tools, posters).

June 26 - July 02 21st **International Conference on Computer Aided Verification** (CAV'2009), Grenoble, France. Topics include: Algorithms and tools for verifying models and implementations, Program analysis and software verification, Verification techniques for security, Applications and case studies, Verification in industrial practice, etc. Deadline for submissions: January 15, 2009 (abstracts), January 25, 2009 (papers, CAV Award nominations).

☺ June 29 - July 03 47th **International Conference Objects, Models, Components, Patterns** (TOOLS Europe'2009), Zurich, Switzerland. Topics include: all aspects of object technology and neighboring fields, in particular model-based development, component-based development, and patterns (design, analysis and other applications); more generally, any contribution addressing topics in advanced software technology; contributions showcasing applications along with a sound conceptual contribution are particularly welcome. Deadline for submissions: January 15, 2009 (technical papers), February-March, 2009 (tutorials, workshops).

☺ June 29 – July 03 9th **International Conference on New Technologies of Distributed Systems** (NOTERE'2009), Montreal, Canada. Deadline for submissions: January 10, 2009.

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. Deadline for paper submissions: January 4, 2009.

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. Deadline for submissions: February 10, 2009.

☺ 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 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. Deadline for paper submissions: February 2, 2009.

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. Deadline for submissions: February 20, 2009 (abstracts), Feburary 27, 2009 (papers).

August 10-13 — 28th Annual ACM SIGACT-SIGOPS **Symposium on Principles of Distributed Computing** (PODC'2009), Calgary, Alberta, Canada.

August 25-28 — 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 submissions: March 2, 2009 (workshops), March 16, 2009 (papers), May 4, 2009 (doctoral symposium abstracts), June 5, 2009 (demos, posters).

☺ 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. Deadline for submissions: January 24, 2009 (abstracts), January 31, 2009 (full papers).

☺ 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. Deadline for submissions: January 20, 2009 (full papers).

Aug 31 – Sep 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.

☺ 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 submissions: February 28, 2009 (abstracts), March 31, 2009 (mini-symposia proposals).

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. Deadline for submissions: March 31 (papers).

☺ September 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. Deadline for submissions: March 20, 2009 (abstracts), March 27, 2009 (papers, tutorials, workshops).

Sepember 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. Deadline for submissions: April 3, 2009 (abstracts), April 10, 2009 (papers), April 28, 2009 (tutorials).

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. Deadline for submissions: March 16, 2009 (papers, tutorials).

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. Deadline for submissions: April 26, 2009 (abstracts), May 10, 2009 (papers).

♦ Oct 07-09 14th **International Real-Time Ada Workshop** (IRTAW'2009), Portovenere, Italy. Deadline for submissions: May 8, 2009 (position papers).

♦ Nov 01-05 2009 ACM **SIGAda Annual International Conference** (SIGAda'2009), Tampa Bay area, Florida, USA. Sponsored by ACM SIGAda, in cooperation with SIGCAS, SIGCSE, SIGPLAN, Ada-Europe, and Ada Resource Association (ACM approval pending; Cooperation approvals pending).

☺ November 02-03 14th **International ERCIM 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. Deadline for submissions: April 1, 2009 (abstracts), April 7, 2009 (papers).

November 02-07 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. Deadline for submissions: December 22, 2009 (workshops), May 4, 2009 (papers).

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

Preliminary Call for Participation
Ada Developer Room at FOSDEM 2009
7-8 February 2009, Brussels, Belgium

FOSDEM[1], the Free and Open source Software Developers' European Meeting, is a free and non-commercial two-day annual event organized in Brussels, Belgium. The 2009 edition will take place on Saturday 7 and Sunday 8 February, 2009. Ada-Belgium[2] organizes a series of presentations related to Ada and Free Software, to be held in a Developer Room on both days of the event.

Preliminary overview:

- An Introduction to Ada for Beginning or Experienced Programmers, *by Jean-Pierre Rosen, Adalog*
- The Object-Oriented Programming Model in Ada 2005, *by Jean-Pierre Rosen, Adalog*
- Ast2Cfg - A Framework for Control Flow Graph Based Analysis and Visualisation of Ada Programs, *by Georg Kienesberger, Vienna University of Technology*
- Ada Annex E - Distributed Systems, *by Thomas Quinot, AdaCore*
- NARVAL - Distributed Data Acquisition from Particle Accelerators, *by Xavier Grave, Centre National de la Recherche Scientifique*
- GPRBuild - A New Build Tool for Large-Scale Software Development, *by Vincent Celier, AdaCore*
- GPS - The GNAT Programming Studio, *by Vincent Celier, AdaCore*
- GNATBench - Ada programming with Eclipse, *by Vincent Celier, AdaCore*
- Ada in Debian, *by Ludovic Brenta, Debian*
- MaRTE-OS - A Hard Real-Time Operating System for Embedded Devices, *by Miguel Telleria de Esteban, Universidad de Cantabria*

The full list with abstracts of presentations and biographies of speakers is available on the Ada at FOSDEM 2009 web-page. More details, such as the concrete schedule will follow later.

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

The FOSDEM Team of Ada-Belgium

---

1 http://www.fosdem.org

2 http://www.cs.kuleuven.be/~dirk/ada-belgium

Call for Industrial Presentations
14th International Conference on Reliable Software Technologies - Ada-Europe 2009
8-12 June 2009, Brest, France

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

Conference Chair

Frank Singhoff
UBO/LISyC, France
Frank.Singhoff@univ-brest.fr

Program Co-Chairs

Yvon Kermarrec
Télécom Bretagne, France
Yvon.Kermarrec@telecom-bretagne.eu

Fabrice Kordon
University Pierre & Marie Curie, France
Fabrice.Kordon@lip6.fr

Tutorial Chair

Jérôme Hugues
Télécom Paris-Tech, France
Jerome.Hugues@telecom-paristech.fr

Exhibition Chair

Pierre Dissaux
Ellidiss Technologies
Pierre.Dissaux@ellidiss.com

Publicity Chair

Dirk Craeynest
Aubay Belgium & K.U.Leuven, Belgium
Dirk.Craeynest@cs. kuleuven.be

Local Chairs

Alain Plantec and Mickael Kerboeuf
UBO/LISyC, France
Alain.Plantec@univ-brest.fr
Mickael.Kerboeuf@univ-brest.fr

Industrial Committee

Guillem Bernat, Rapita Systems, UK
Agusti Canals, CS, France
Roderick Chapman, Praxis HIS, UK
Colin Coates, Telelogic, UK
Dirk Craeynest, Aubay Belgium & K.U.Leuven, Belgium
Dirk Dickmanns, EADS, Germany
Tony Elliston, Ellidiss Software, UK
Franco Gasperoni, AdaCore, France
Hubert Keller, Forschungszentrum Karlsruhe GmbH, Germany
Bruce Lewis, US Army, USA
Ahlan Marriott, White-Elephant GmbH, Switzerland
Rei Strähle, Saab Systems, Sweden

In cooperation with
ACM SIGAda

General Information

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

Call for Industrial Presentations

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

12 January 2009	Submission of industrial presentation proposals
09 February 2009	Notification to all authors
11 May 2009	Industrial presentations required
8-12 June 2009	Conference

Call for Exhibitions

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

Topics

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

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

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

14TH INTERNATIONAL REAL-TIME ADA WORKSHOP
IRTAW-14

7-9 October 2009
Portovenere
Italy
http://events.math.unipd.it/irtaw14/

CALL FOR PAPERS

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.

In keeping with this tradition, and in light of Ada 2005 implementations beginning to appear, and thought of post Ada 2005 language changes, the goals of **IRTAW-14** will be to:

- examine experiences in using Ada 2005 for the development of real-time systems and applications;
- report on or illustrate implementation approaches for the real-time features of Ada 2005;
- consider the added value of developing other real-time Ada profiles in addition to the Ravenscar profile;
- examine the implications to Ada of the growing use of multiprocessors in the development of real-time systems, particularly with regard to predictability, robustness, and other issues;
- examine and develop paradigms for using Ada 2005 for real-time distributed systems, taking into account robustness as well as hard, flexible and application-defined scheduling;
- consider the definition of specific patterns and libraries for real-time systems development in Ada;
- identify how Ada relates to the certification of safety-critical and/or security-critical real-time systems;
- review the status and contents of ISO reports related to real-time Ada and consider the interest of developing new secondary standards or extensions;
- examine the status of the Real-Time Specification for Java and other languages for real-time systems development, and consider user experience with current implementations and with issues of interoperability with Ada in embedded real-time systems;
- consider the lessons learned from industrial experience with Ada and the Ravenscar Profile in actual real-time projects;
- consider the language vulnerabilities of the Ravenscar and full language definitions.

Participation at **IRTAW-14** is by invitation following the submission of a position paper addressing one or more of the above topics or related real-time Ada issues. Alternatively, anyone wishing to receive an invitation, but for one reason or another is unable to produce a position paper, may send in a one-page position statement indicating their interests. Priority will, however, be given to those submitting papers.

Position papers should not exceed ten pages in typical IEEE conference layout, excluding code inserts. All accepted papers will appear, in their final form, in the Workshop Proceedings, which will be published as a special issue of *Ada Letters* (ACM Press). Selected papers will also appear in the Ada User Journal.

Please submit position papers, in PDF format, to the Program Chair by e-mail: **neil@cs.york.ac.uk**

Program Committee

Neil Audsley (Program Chair), Ben Brosgol, Alan Burns, Michael González Harbour, Stephen Michell, Javier Miranda, Luís Miguel Pinho, Juan Antonio de la Puente, Jorge Real, José Ruiz, Tullio Vardanega (Local Chair) and Andy Wellings.

Important Dates

Receipt of Position Paper:	**8 May 2009**
Notification of Acceptance:	**22 May 2009**
Final Copy of Paper:	**16 September 2009**
Workshop Date:	**7-9 October 2009**

Call for Technical Contributions – SIGAda 2009

ACM Annual International Conference
on Ada and Related Technologies:
Engineering Safe, Secure, and Reliable Software

Hilton St. Petersburg Bayfront Hotel
Tampa Bay, Florida, USA
November 1-5, 2009

Submission Deadline: June 30, 2009
Sponsored by ACM SIGAda
http://www.acm.org/sigada/conf/sigada2009

SUMMARY: Reliability, safety, and security are among the most critical requirements of contemporary software. The application of software engineering methods, tools, and languages all interrelate to affect how and whether these requirements are met.

Such software is in operation in many domains of application. Much has been accomplished in recent years, but much remains to be done. Our tools, methods, and languages must be continually refined; our management process must remain focused on the importance of reliability, safety, and security; our educational institutions must fully integrate these concerns into their curricula.

The conference will gather industrial and government experts, educators, software engineers, and researchers interested in developing, analyzing, and certifying reliable, safe, secure software. We are soliciting technical papers and experience reports with a focus on, or comparison with, Ada.

We are especially interested in experience in integrating these concepts into the instructional process at all levels.

POSSIBLE TOPICS INCLUDE BUT ARE NOT LIMITED TO:

- Transitioning to Ada 2005
- Challenges for developing reliable, safe, secure software
- Ada and SPARK in the classroom and student laboratory
- Language selection for highly reliable systems
- Mixed-language development
- Use of high reliability subsets or profiles such as MISRA C, Ravenscar, SPARK
- High-reliability standards and their issues
- Software process and quality metrics
- System of Systems
- Real-time networking/quality of service guarantees
- Analysis, testing, and validation
- Use of ASIS for new Ada tool development
- High-reliability development experience reports
- Static and dynamic analysis of code
- Integrating COTS software components
- System Architecture & Design
- Information Assurance
- Ada products certified against Common Criteria / Common Evaluation Methodology
- Distributed systems
- Use of new Ada 2005 features/capabilities
- Fault tolerance and recovery
- Performance analysis

KINDS OF TECHNICAL CONTRIBUTIONS:

TECHNICAL ARTICLES present significant results in research, practice, or education. Articles are typically 10-20 pages in length. These papers will be double-blind refereed and published in the Conference Proceedings and in ACM Ada Letters. The Proceedings will be entered into the widely-consulted ACM Digital Library accessible online to university campuses, ACM's 80,000 members, and the software community.

EXTENDED ABSTRACTS discuss current work for which early submission of a full paper may be premature. If your abstract is accepted, you will be expected to produce a full paper, which will appear in the proceedings. Extended abstracts will be double-blind refereed. In 5 pages or less, clearly state the work's contribution, its relationship with previous work by you and others (with bibliographic references), results to date, and future directions.

EXPERIENCE REPORTS present timely results on the application of Ada and related technologies. Submit a 1-2 page description of the project and the key points of interest of project experiences. Descriptions will be published in the final program or proceedings, but a paper will not be required.

PANEL SESSIONS gather a group of experts on a particular topic who present their views and then exchange views with each other and the audience. Panel proposals should be 1-2 pages in length, identifying the topic, coordinator, and potential panelists.

WORKSHOPS are focused work sessions, which provide a forum for knowledgeable professionals to explore issues, exchange views, and perhaps produce a report on a particular subject. A list of planned workshops and requirements for participation will be published in the Advance Program. Workshop proposals, up to 5 pages in length, will be selected by the Program Committee based on their applicability to the conference and potential for attracting participants.

TUTORIALS offer the flexibility to address a broad spectrum of topics relevant to Ada, and those enabling technologies which make the engineering of Ada applications more effective. Submissions will be evaluated based on relevance, suitability for presentation in tutorial format, and presenter's expertise. Tutorial proposals should include the expected level of experience of participants, an abstract or outline, the qualifications of the instructor(s), and the length of the tutorial (half-day or full-day). Tutorial presenters receive complimentary registration to the other tutorials and the conference.

HOW TO SUBMIT: Send contributions by **June 30, 2009**, in Word, PDF, or text format as follows**:**

Technical Articles, Extended Abstracts, Experience Reports, and Panel Session Proposals: Program Chair, Lt. Col. Jeff Boleng (Jeff.Boleng@usafa.edu)

Workshop Proposals: Workshops Chair, Bill Thomas (BThomas@mitre.org)

Tutorial Proposals: Tutorials Chair, Richard Riehle (RDRiehle@nps.edu)

FURTHER INFORMATION:

CONFERENCE GRANTS FOR EDUCATORS: The ACM SIGAda Conference Grants program is designed to help educators introduce, strengthen, and expand the use of Ada and related technologies in school, college, and university curricula. The Conference welcomes a grant application from anyone whose goals meet this description. The benefits include full conference registration with proceedings and registration costs for 2 days of conference tutorials/workshops. Partial travel funding is also available from AdaCore to faculty and students from GNAT Academic Program member institutions, which can be combined with conference grants. For more details visit the conference web site or contact Prof. Michael B. Feldman (mfeldman@gwu.edu).

OUTSTANDING STUDENT PAPER AWARD: An award will be given to the student author(s) of the paper selected by the program committee as the outstanding student contribution to the conference.

SPONSORS AND EXHIBITORS: Please contact Alok Srivastava (Alok.Srivastava@auatac.com) for information about becoming a sponsor and/or exhibitor at SIGAda 2009.

IMPORTANT INFORMATION FOR NON-US SUBMITTERS: International registrants should be particularly aware and careful about visa requirements, and should plan travel well in advance. Visit the conference website for detailed information pertaining to visas.

ANY QUESTIONS?:

Please submit your questions on the conference to the Conference Chair, Greg Gicca (gicca@adacore.com) or Local Arrangements Chair Currie Colket (colket@acm.org).

Ada-C++ Interfacing in the ERAM System

Howard Ausden

Lockheed Martin Corp, 9211 Corporate Boulevard, Rockville, Maryland 20850, USA; Tel: +1 301 640 2099; email: howard.ausden@lmco.com

Abstract

This paper describes the approach used in the ERAM air traffic control system to integrate flight objects, published by Ada code, with C++ client software bound into the same executable.

Keywords: Ada, C++, multi-language.

1 Introduction

The En Route Automation Modernization (ERAM) program is a real-time Air Traffic Control (ATC) program being developed by Lockheed Martin Corporation. ERAM is a United States FAA program: see http://www.faa.gov/airports_airtraffic/technology/eram.

ERAM has high availability requirements, mission critical applications, and stringent response time requirements. The estimated size of the ERAM system is 1,300K statements of primarily Ada and C++ code.

This paper discusses some advantages and challenges encountered in the development of the interface for passing flight data between Ada and C++ components of ERAM.

2 Software Architecture for Flight Data

The operational ERAM software consists of executables running on UNIX, communicating by passing messages. The flight server is one such executable, and is written entirely in Ada. It runs on a central server, processing incoming messages to create, update, and delete flights, and publishing flight data messages to client executables. Each client executable links in libraries containing the methods it needs to interact with the flight object. The majority of the dozen or so client executables are written in Ada; however, some of the client executables are written in C++. The C++, more than 150K statements, was reused from an earlier system because it had years of investment that made it a fully tested and operationally suitable product, and the customer liked it. This C++ code runs in the controller workstations and displays flight data, so it needs to call many of the flight object's methods.

The software design choices were to:

- Re-write the Ada client-side libraries in C++ (the libraries receive the published flight data, maintain a flight database, and provide methods for clients to interact with flight data)
- "Wrap" the Ada methods with C++.

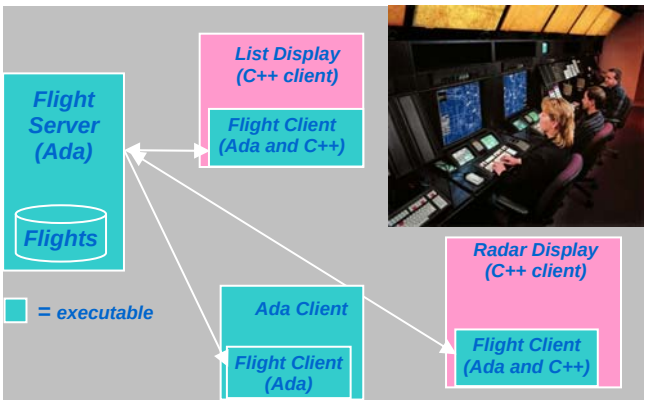


Figure 1 Simplified architecture for flight data

Because some flight object methods are long and complex (for example, the routines to predict position of an aircraft) it seemed better to implement wrapper code, rather than writing and forever maintaining both Ada and C++ versions.

Colleagues who had written cross-language code previously encouraged us to keep the cross-language API simple; they suggested using simple types like integer, Boolean, and records, that could be exchanged directly between the languages. This seemed impossible when considering the size and complexity of the ERAM flight object (shown below in overview).

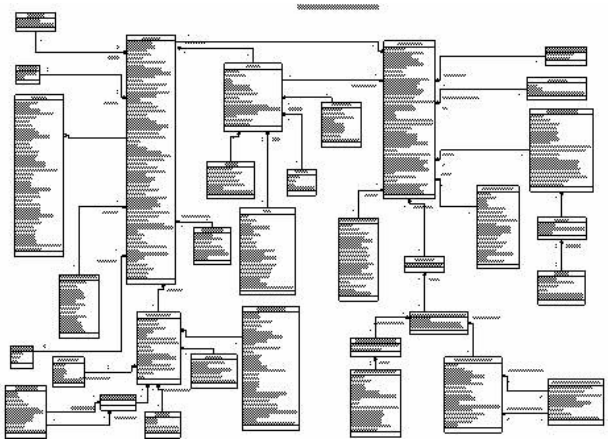


Figure 2 ERAM flight object model

3 Ada – C++ Interfacing

3.1 API Generation

Our initial approach was to develop a tool to generate the C++ header files from the Ada specs (described in reference [1]). The intent was to ensure consistency and to save effort; however:

1. Ensuring consistency between Ada and C++ can only be done correctly at build time, since somebody might edit the generated API before the system is built.
2. The generated API was unnatural; it did not employ types or conventions that were natural to the language. Since many developers would write code against the API, it was important that it should employ the best native techniques.
3. While the tool did help with the initial generation of the C++ API, subsequent maintenance typically involves minor updates such as adding a parameter to a method. In this case, a code generator increases, rather than saves, work.

3.2 General Wrapper Approach

In the case of our flight data libraries, the C++ client is generally interrogating the flight object. Figure 3 below shows the general approach.

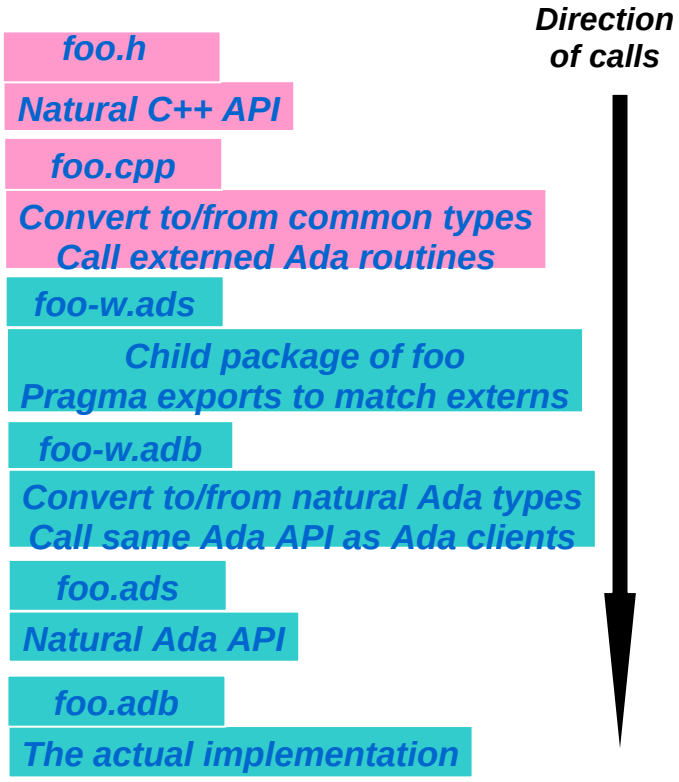


Figure 3 Wrapper approach for C++ calling Ada

In both languages, there is a natural API that does not reflect concerns with passing data between the languages. In the C++ code, data is moved into types that are simple enough to exchange between the languages, and externed Ada routines are called. Data items returned from Ada are constructed into C++ objects.

On the Ada side, the package with the natural Ada API is not usually called directly from C++; the wrapper functionality is encapsulated in a child package (the file naming convention is <parent file>-w for wrappers). The child package converts the intermediate types into native Ada types and calls the natural Ada API like any other Ada client would. Reference [2] contains an example of this wrapper approach.

3.3 Cross-Language Consistency Checking

Of course the Ada and C++ are compiled separately yet they are linked together in the same executable. The consistency of bit layouts between the Ada and C++ types is ensured by running a type matching tool at build time. This is described in reference [3].

We are currently investigating the feasibility of a tool to ensure that method signatures match, since the linker matches by method name only. Since Ada requires methods called in/from another language to be specified in a pragma import/export, the tool will scan the source code to find the pragma imports and exports in the Ada, and will then search for matching calls in the C++.

3.4 Cross-Language Data Types

ERAM and its ancestor systems use types dictionaries to allow online and offline applications to know information about types like bit layouts (see reference [4]). Initially we used a type generator tool to create both Ada and C++ types with identical bit layouts from the dictionary. However the types were unnatural (for example, they contained padding fields) and code had to be written in both languages to move data in and out of the unnatural types. As developers gained expertise they learned to design types that align, or at least to design an intermediate type that matches the natural type in one of the languages, thus saving both labor and runtime cycles to copy data in and out of intermediate objects. For example, booleans are 32 bits in C++ and 8 bits in Ada; by writing a representation clause to force the Ada compiler to use 32 bits, the type can be directly exchanged with C++.

3.5 Large, Complex Types

Only simple types with identical bit layouts can be passed between languages. However, as mentioned earlier the ERAM flight object is highly complex and contains many complex sub-objects. When passing a large, complex type between languages the rule is to divide and conquer. Break the large type into small pieces: make multiple calls and fetch piecemeal; construct the complex object in the other language. The smaller the piece, the easier it is to make the types align. **Arbitrarily large, complex types can be passed this way.**

3.5.1 Repeated Elements

There are two ways to pass repeated elements (for example, an array/table or a vector) between languages.

- Pass the number of items and then loop and pass each element, or
- Pack the elements into a buffer:

- Pack attributes and type IDs into the buffer in one language
- Pass the buffer address to the other language
- Unpack the attributes; use the type ID to move bytes into the native type.

3.6 Performance Considerations

ERAM is a real-time system. The C++ display code needs to interact with the Ada flight object many times in one second. Of course the sub-objects within a flight such as the flight plan, route, and trajectory, are represented by private types in both languages. This allows the internal implementation to employ performance optimizations. When a C++ client gets a sub-object from a flight, the wrapper design depends on the complexity of the type and the client's calling pattern:

- The whole object may be fetched to C++ (slow) and then individual 'get attribute' functions will be pure C++ (fast)
- Or, just the object's handle is passed across languages (fast); methods are wrappers of Ada routines (slow); once fetched to the C++ side, attributes are cached so subsequent access is fast.

Complex methods such as position prediction are implemented as wrappers of Ada routines, regardless of whether the whole object was passed to C++.

3.6.1 Smart Pointer Wrapper Example

Since a flight object is constant until next updated by the server, multiple readers can share one copy. This makes a smart pointer a good choice for a handle for a flight object or sub-object.

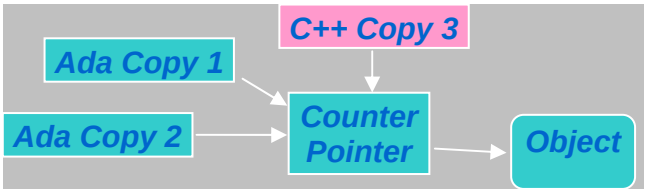


Figure 4 Smart pointer with Ada and C++ clients

A smart pointer consists of a pointer to an object and a reference count. Smart pointers avoid memory leaks by counting references (i.e., pointers) to an object and freeing the memory when the reference count reaches zero. They also avoid the cost of copying, since all readers point to a single object.

A C++ wrapper for the smart pointer allows these benefits to be enjoyed by both C++ and Ada code. In Ada, the reference count is maintained by the Adjust and Finalize routines (see Ada.Finalization.Controlled). In C++, the client must overload the assignment and destructor operations and make calls to the Ada copy and finalize routines.

3.7 Calling into C++ from Ada

The foregoing discussion describes how C++ can fetch data from Ada; how about when Ada needs to call C++? In ERAM, this situation arises when a flight is received from the server and the displays must be updated. The C++ client has a flight listener object with an accept_update routine that must be dispatched whenever a flight update is received by the Ada code. But how can Ada code dispatch a C++ method?

To solve the problem, an Ada listener object is created that contains a pointer to the C++ object. The Ada listener is registered with the Ada API like any other Ada client's listener. Any calls to the Ada listener object are passed to C++ via static/non-dispatching routines (visible to Ada via pragma import) that take the C++ object pointer as a parameter. The static C++ routine can then perform the normal object call and dispatch the routine. This simple and powerful technique allows us to use object-oriented design across the languages.

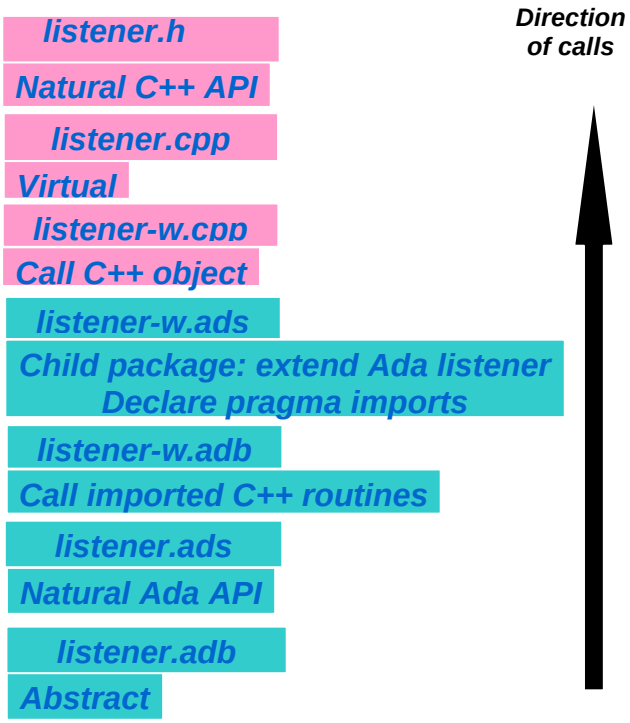


Figure 5 Calling into C++ from Ada

The approach is shown in the figure above:

- The package listener.ads contains a tagged listener type and an abstract Accept_Update method
- The child package listener-w.ads extends the type with a pointer to the C++ listener, and implements Accept_Update to call the imported C++ accept_update routine
- The C++ wrapper listener-w.cpp inherits from the listener type in listener.h, and calls the C++ user's implementation of accept_update.

3.8 Exception Propagation across Languages

What happens when an exception is propagated in one language, and reaches the boundary between the languages? If the exception is handled in the other language, control will return to the original language but the stack will not have been unwound and the program will crash. This is because each language has its own runtime management software.

Either of two approaches can be used:

- Catch all exceptions in wrapper routines, pass a return code corresponding to the exception to the other language, and then raise a corresponding exception in that language
- Or, catch all exceptions and call a utility routine provided by our compiler vendor. The utility routine unwinds the stack before calling the other language and raising a corresponding exception.

4 Conclusion

In the ERAM flight application the wrappers contain some 10K statements of C++ and a similar amount of Ada, yet over the life of the ERAM system the wrapper approach will be cheaper than maintaining duplicate C++ and Ada versions of the client-side flight code.

Thanks to the strengths of Ada, such as representation clauses and pragma import/export, every cross-language problem was solved, including passing large, complex objects, and object-oriented programming. These techniques have worked successfully through several levels of testing over more than 2 years, and will be in operational use before the end of 2009.

References

[1] Howard Ausden and Karl Nyberg (November 2005), *Using ASIS to generate C++ bindings*, ACM SIGAda Ada Letters, Proceedings of the 2005 annual ACM SIGAda international conference on Ada: The Engineering of Correct and Reliable Software for Real-Time & Distributed Systems using Ada and Related Technologies SigAda '05, Volume XXV Issue 4

[2] Howard Ausden (June 2008), *Ada-C++ Interfacing in the Flight Services Component of the ERAM System*, presentation at Ada-Europe 2008

[3] Matt Mark (November 2005), *Data sharing between Ada and C/C++*, ACM SIGAda Ada Letters, Proceedings of the 2005 annual ACM SIGAda international conference on Ada: The Engineering of Correct and Reliable Software for Real-Time & Distributed Systems using Ada and Related Technologies SigAda '05, Volume XXV Issue 4

[4] Michael Glasgow, Donna Hepner, Richard Schmidt (1991), *Implementing a table-driven types dictionary service in Ada*, paper presented at EUROSPACE Ada in AEROSPACE Symposium, November 1991. Unpublished

Porting Naval Command & Control Systems to Ada 2005

Jeff Cousins

BAE Systems Integrated System Technologies (Insyte) Limited KT3 4LH; UK; Tel: +44 20 8329 5430; email: jeff.cousins@baesystems.com

Abstract

This paper describes our experience of porting large naval Command and Control Systems from Ada 95 to Ada 2005. It covers: the area of application; what code changes did Ada 2005 require;and new features and tools with Ada 2005.

1 Introduction

This paper discusses the impact of the Ada 2005 language changes upon a family of naval Command and Control Systems. It describes which language changes have had the most impact, and what changes were necessary to our code to make it Ada 95 - 2005 portable.

The paper begins with an overview of the area of application. This is a family of naval Command and Control Systems developed by BAE Systems Insyte over many years. Extensive use is made of the Ada language. Legacy code had been ported previously from Ada 83 to Ada 95.

Our naval projects predominantly use the GNAT compiler. We have generally adopted new versions of GNAT within a few months of their release, after an evaluation period, so as to pick up fixes and other benefits such as more efficient back ends.GNAT has progressively added the new features of Ada 2005, and most of the new language was available (with the use of an Ada 2005 compiler switch) even before the language was finally published by ISO, thus allowing an early evaluation of the impact.

One of the goals of the 1995 and 2005 revisions of the Ada language was to maintain backward compatibility with the previous version. This did not however preclude making changes where it was thought that there would be a worthwhile improvement in the safety and security of the language. Thus work has been necessary to bring legacy code up to date to be Ada 2005 compliant.

Although the Ada 2005 compiler switch has not yet been used for customer deliveries, we have attempted to achieve a degree of future proofing by writing our code to be portable across both Ada 95 and Ada 2005.

Many of the recent language changes were classified as retrospective corrections to Ada 95, so we were impacted by the changes even without using the Ada 2005 switch.

The paper identifies each language change that has required a code change, what the scale of the impact was, and what the code change was. Sometimes alternative solutions were available, one for a pure Ada 2005 system, and another that was less neat but which was backwardly compatible with Ada 95. The paper compares which changes caused "incompatibilities" detectable at compile time, and which caused "inconsistencies" not detected until execution.

The biggest impact came from the introduction of new reserved words, in particular “Interface”. Although the solution was a very simple name change, it was quite hard to manage. A package called Interface is by its nature likely to be on a boundary between subsystems, so a name change will have widespread affect, and different projects may be ready to adopt the change at different times. Most other changes were localised to a few files. Overall, the work necessary to port from Ada 95 to 2005 has been found to be much less than that incurred in porting from Ada83 to 95.

Although our changes have mostly been a porting exercise, some limited use has been made of the new features, such as Vectors from the Containers package, the Environment_Variables interface, and pragma Restrictions.

Besides the language changes, the paper mentions other new utilities that can be used to improve the reliability of Ada programs. The GNAT compiler now provides additional utilities for dynamic stack usage measurement and for the detection of un-initialised variables. These have proved useful for drawing attention to potential problems.

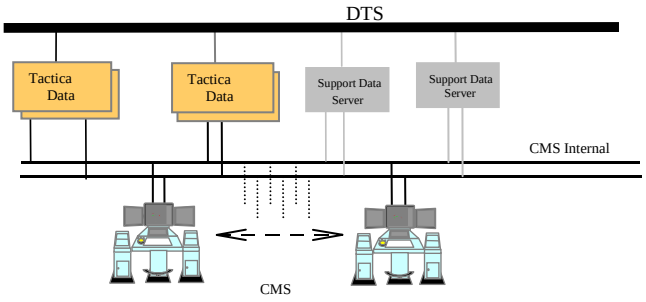
2 Area of application

CMS-1 family of Command Systems

The CMS-1 family of Command Systems was initially developed for the Royal Navy’s Type 45 destroyers, inheriting legacy code from the Type 23 frigates and Type 42 destroyers. It has also been fitted to the Falklands patrol vessel HMS Clyde, the helicopter training and casualty reception ship RFA Argus, and is in development for retro-fit to the Type 23 frigates and for fitting to the CVF carriers.

CMS-1 systems have dual redundant Servers hosting software written mostly in Ada: the Type 45 variant has 1.2 million lines of Ada and the Type 23 variant 1 million lines of Ada. This is contained in 19 thousand original Ada source files, 85 thousand after code generation. CMS-1 systems have multiple Multi-Function Consoles for operator interaction, hosting software written mostly in C++ (about 350 thousand lines).

Basic CMS-1 Architecture



The architecture is a conventional one comprising dual redundant servers connected to multiple operator consoles by a dual redundant LAN, and to ship's equipments by a dual or triple redundant ship's highway (DTS in this case).

3 What code changes did Ada 2005 require?

- There are new Reserved Words which can no longer be used as identifiers.
- 'Access is no longer allowed in generic bodies.
- Limited function results may require explicit use of an access type.
- Explicit null exclusion may be required for renamed subprograms.
- There is an inconsistency in Interfaces.C.Strings.

History of porting: Ada 83 to Ada 95

Much of the code had previously been ported from Ada 83 to 95. For example, Ada 95 has a single exception handler for Constraint and Numeric Errors, so code was ported from:

```
when CONSTRAINT_ERROR =>
  - - Handler;
when NUMERIC_ERROR =>
  - - Handler;
```

to:

```
when CONSTRAINT_ERROR | NUMERIC_ERROR=>
  - - Common handler;
```

Ada 95 to 2005 porting required only about 10% of the effort of that Ada 83 to 95 porting had.

3.1 New reserved words

The biggest impact of all was due to the word "interface" becoming a reserved word in Ada 2005.

A fairly common model for structuring packages is to a have a minimal parent package X, with various child packages hanging off of it, one of which publishes the user interface. Naturally enough this child package would usually be called X.Interface. Also, occasionally variables were called Interface.

A convention of simply adding a prefix seemed to have already developed in the wider Ada world, e.g. GC_Interface for GNATCOM.

Items named using the traditional "English" spelling synchronised were fortunately not affected by the "American" (and Oxford English Dictionary) synchronized becoming reserved.

This affected 149 files. (The number of files is given rather than the number of lines since the editing time was small. Files are the unit of configuration control, and it was the getting of files from the configuration management system and putting them back that took the time).

Although the editing time was small, an Interface child package was by its nature likely to be on the interface between different teams, so some co-ordination was required, and it took some months before all the teams affected were ready to take the change.

3.2 No 'Access in generic bodies

'Access is no longer allowed in generic body (AI-229).

There was already an established work-around of adding a constant set to x'Access to private part of the spec, and then using this constant in the body.

This was a retrospective change to Ada 95, besides being required for Ada 2005, so if you are using a recent compiler you may have to make code changes even if you are not using any new Ada 2005 features.

This affected 16 files.

In the following example, a generic spec declares a call-back routine. In the generic body, one of its procedures calls a procedure in a second package, passing the call-back routine as a parameter.

Original Ada 95

```
generic
   - - Parameters
package P is
   - - Types
   - - Subprogram specs including
   procedure Callback;
private
   - - Types
   - - Subprogram specs
end P;

 - - Context clauses including
with Y;
generic
package body P is
   - - Subprogram bodies including
   procedure Callback is
   begin
      - -;
   end Callback;

   procedure Proc is
   begin
      - -
      Y.Z (
       Callback_Ptr => Callback'Access);
```

```
      - -
   end Proc;
end P;
```

New Ada 95 and Ada 2005

```
- - Context clauses including
with X;
generic
   - - Parameters
package P is
   - - Types
   - - Subprogram specs including
   procedure Callback;
private
   - - Types
   - - Subprogram specs
   Callback_Access : constant
                X.Callback_Ptr_Type :=
                Callback'Access
end P;

- - Context clauses including
with Y;
generic
package body P is
   - - Subprogram bodies including
   procedure Callback is
   begin
      - -;
   end Callback;

   procedure Proc is
   begin
      - -
      Y.Z (
       Callback_Ptr => Callback_Access);
      - -
   end Proc;
end P;
```

3.3 Limited function results

Ada 2005 has dropped the concept of implicit pass by-reference types, functions returning a value of a limited type must explicitly return by access type.

The workaround is to declare the objects to be returned as aliased, return an access type pointing to them, and let caller de-reference using .all.

Using an Ada 2005 anonymous access type would affect the fewest files but then the code would not be Ada 95/2005 portable. Having language specific variants would be a pain for configuration control, so a named access type has been used.

This affected 6 files.

In the following example, the first package T.GF declares a limited private type for use as a "handle" type for pointing to objects. The second package R.MC declares a selector function that returns the appropriate handle depending on some parameter. The third package R.MGI declares a pool of handles. The body of R.MC gives the body of the function for choosing a handle. At the end is a call of the selector function.

Original Ada 95

```
package T.GF is
   type Gsi_Handle_Type is limited
                              private;
   - - Other types
   - - Subprogram specs
private
   type GSI_Handle_Type is …;
end T.GF;

with T.GF;
with T.S;
package R.MC is
   - - Types
   - - Subprogram specs including
   function Get_GSI_Handle (
    Source_Name : in T.S.Source_Name_T)
    return          T.GF.GSI_Handle_Type;
end R.MC;

package R.MGI is
    - - Subprogram specs
    GSI_Handle_A :
                    T.GF.GSI_Handle_Type;
    GSI_Handle_B :
                    T.GF.GSI_Handle_Type;
end R.MGI;

with R.MGI;
package body R.MC is
   - - Subprogram bodies including
   function Get_GSI_Handle (
    Source_Name : in T.S.Source_Name_T)
    return        T.GF.GSI_Handle_Type is
   begin
     case Source_Name is
       when A =>
         return R.MGI.GSI_Handle_A;
       when B =>
         return R.MGI.GSI_Handle_B;
     end case;
   end Get_GSI_Handle;
end R.MC;

GSI_Handle := R.MC.Get_GSI_Handle
                        (My_Source_Name);
```

Ada 2005 only

Package T.GF is unchanged.

```
with T.GF;
with T.S;
package R.MC is
   - - Types
   - - Subprogram specs including
   function Get_GSI_Handle (
    Source_Name : in T.S.Source_Name_T)
    return access T.GF.GSI_Handle_Type;
end R.MC;
```

```
package R.MGI is
   - - Subprogram specs
   GSI_Handle_A :
         aliased T.GF.GSI_Handle_Type;
   GSI_Handle_B :
         aliased T.GF.GSI_Handle_Type;
end R.MGI;

with R.MGI;
package body R.MC is
  - - Subprogram bodies including
  function Get_GSI_Handle (
   Source_Name : in T.S.Source_Name_T)
   return access T.GF.GSI_Handle_Type
   is
  begin
    case Source_Name is
      when A =>
        return
            R.MGI.GSI_Handle_A'Access;
       when B =>
        return
            R.MGI.GSI_Handle_B'Access;
    end case;
  end Get_GSI_Handle;
end R.MC;

GSI_Handle := R.MC.Get_GSI_Handle
                (My_Source_Name).all;
```

Ada 95/2005 portable

```
package T.GF is
  type Gsi_Handle_Type is limited
                          private;
  type Gsi_Handle_Type_Ptr is access
                all Gsi_Handle_Type;
  - - Other types
  - - Subprogram specs
private
  type GSI_Handle_Type is …;
end T.GF;

with T.GF;
with T.S;
package R.MC is
  - - Types
  - - Subprogram specs including
  function Get_GSI_Handle (
   Source_Name : in T.S.Source_Name_T)
   return T.GF.GSI_Handle_Type_Ptr;
end R.MC;

package R.MGI is
   -- Subprogram specs
   GSI_Handle_A :
         aliased T.GF.GSI_Handle_Type;
   GSI_Handle_B :
         aliased T.GF.GSI_Handle_Type;
end R.MGI;

with R.MGI;
package body R.MC is
  - - Subprogram bodies including
  function Get_GSI_Handle (
   Source_Name : in T.S.Source_Name_T)
   return T.GF.GSI_Handle_Type_Ptr
   is
  begin
    case Source_Name is
      when A =>
        return
            R.MGI.GSI_Handle_A'Access;
       when B =>
        return
            R.MGI.GSI_Handle_B'Access;
    end case;
  end Get_GSI_Handle;
end R.MC;

GSI_Handle := R.MC.Get_GSI_Handle
                (My_Source_Name).all;
```

3.4 Explicit null exclusion for renamed subprograms

When a subprogram renames another and they dispatch upon an access parameter, then the access parameter must be explicitly null excluding (RM-3.9.2, changed as a result of AI95-00404-01).

It would be easy enough to add the words "not null" but then the code would not be Ada 95/2005 portable. Having language specific variants would be a pain for configuration control, so the renames has been changed to a body calling the first named routine – a stepping stone procedure. This incurs a marginal overhead, but the procedure is only called infrequently.

Note that AI447, once implemented, will retrospectively change Ada 95 to allow explicit null exclusion, to ease transition.

This affected 2 files.

In the following example the first parameter is dispatching.

Original Ada 95

```
procedure Cleared (
  Subscriber          : access  Subscriber_Type;
  First_Instance_Id : in Instance_Id_Type;
  Last_Instance_Id : in Instance_Id_Type
   ) is
begin
   - -;
end Cleared;

procedure Partially_Cleared (
  Subscriber          : access Subscriber_Type;
  First_Instance_Id : in Instance_Id_Type;
```

```
   Last_Instance_Id   : in Instance_Id_Type
   ) renames Cleared;
```

Ada 2005 only

```
procedure Cleared (
   Subscriber       : not null access Subscriber_Type;
   First_Instance_Id : in Instance_Id_Type;
   Last_Instance_Id : in Instance_Id_Type
   ) is
begin
   - -;
end Cleared;

procedure Partially_Cleared (
   Subscriber       : not null access Subscriber_Type;
   First_Instance_Id : in Instance_Id_Type;
   Last_Instance_Id : in Instance_Id_Type
   ) renames Cleared;
```

Ada 95/2005 portable

```
procedure Cleared (
   Subscriber       : access Subscriber_Type;
   First_Instance_Id : in Instance_Id_Type;
   Last_Instance_Id : in Instance_Id_Type
   ) is
begin
   - -;
end Cleared;

procedure Partially_Cleared (
   Subscriber       : access Subscriber_Type;
   First_Instance_Id : in Instance_Id_Type;
   Last_Instance_Id : in Instance_Id_Type
   ) is
begin
   Cleared (
      Subscriber        => Subsciber,
      First_Instance_Id => First_Instance_Id,
      Last_Instance_Id => Last_Instance_Id);
end Partially_Cleared;
```

3.5 Inconsistency in Interfaces.C.Strings

The procedure Update in Interfaces.C.Strings no longer adds a nul character (AI-242).

This was a retrospective change to Ada 95, besides being required for Ada 2005, so if you are using a recent compiler (v5.04 or later for GNAT) you may have to make code changes even if you are not using any new Ada 2005 features.

This only affected 1 file, but it was not a very nice language change since it was not an “incompatibility” detectable by the compiler, but was an “inconsistency” that was not found until run time. (An incompatibility would be reported by the compiler as illegal code, an inconsistency is not illegal but the software no longer behaves as before).

4 New features and tools with Ada 2005

We have already made use of some of the new language features:

- Access to environment variables;
- Containers;
- pragma Restrictions.

Some new tools from the vendor which came along at around the same time as the Ada 2005 Amendment are also discussed:

- GNATStack;
- Dynamic stack usage analysis;
- Initialize_Scalars;
- Warnings for un-initialised out parameters;
- Back-end switch for further searching for un-initialised variables.

4.1 New Features – Access to environment variables

We have already made use of package Environment_Variables. Unfortunately it has separate functions Exists (to test whether an environment variable exists) and Value (to read the value of the environment variable), the latter raising an exception if the environment variable does not exist. It would have been more convenient if Value simply returned a null string if the environment variable did not exist – it is normally regarded as bad practice to use an exception for what may be normal behaviour.

4.2 New Features - Containers

Our tools team have already made use of Vectors for an extensible array to store "lists" of mappings between record component names and enumeration literals. No problems were experienced in using them.

It is probably an implementation issue, but the impression that we had from looking at the GNAT implementation was that Ada Vectors after most operations are exactly the capacity they need to be. They do not provide an extra bit of capacity for expansion in the same way as GNAT's implementation of Ada Unbounded Strings does, so memory needs to be re-assigned every time that the capacity grows. It is easy to work around this by making calls to Reserve_Capacity and overestimating the capacity required, but it is not as convenient.

We still use Booch Unbounded Maps for mapping items such as enumeration literals to their represented values and vice versa. Ada 2005 Hashed_Maps do not appear to provide the bucket statistics that Booch Maps do. Bucket statistics are useful in determining the effectiveness of the hashing function. We can test the hashing function in isolation, but it would be more convenient to be able to get the statistics from the container.

Overall, it is nice to finally have containers as part of the predefined Ada library.

4.3 New Features – pragma Restrictions

The Ravenscar profile is too restrictive for our needs, but we have been experimenting with our own set of pragma Restrictions.

The aim is to limit unauthorised programming styles, and to potentially allow some optimisation of the code generated or the tuning of the run-time required, though in practice little reduction in image size has been found (other than for a couple of GNAT-specific Restrictions).

Immediate_Reclamation	RM H.4(10)
Max_Asynchronous_Select_Nesting => 0	RM D.7(18)
Max_Entry_Queue_Length => 1	RM D.7(19.1/2)
Max_Protected_Entries => 2	RM D.7(14)
Max_Select_Alternatives => 5	RM D.7(12)
Max_Storage_At_Blocking => 0	RM D.7 (17)
Max_Tasks => 50	RM D.7(19/1)
Max_Task_Entries => 3	RM D.7(13)
No_Abort_Statements	RM D.7(5)
No_Dynamic_Attachment	RM D.7(10/2)
No_Dynamic_Priorities	RM D.7(9/2)
No_Implicit_Heap_Allocations	RM D.7(8)
No_Local_Protected_Objects	RM D.9(10.1/2)
No_Nested_Finalization	RM D.7(4/2)
No_Obsolescent_Features	RM 13.12.1(4/2)
No_Requeue_Statements	RM D.7(10.5/2)
No_Task_Termination	RM D.7(15.1/2)
No_Unchecked_Access	RM H.4(18)

No_Nested_Finalization is not possible for programs that make use of GNATCOM for interoperation with Windows COM.

4.4 Other new tools - GNATStack

This is an add-on tool to GNAT. It gives the stack usage for each subroutine so you can look for subroutines that use unusually large amounts of stack. (This information was already available by compiling using the –fstack-usage switch).

It does not follow indirect calls so it cannot work out the worst case stack usage for a task that uses run-time dispatching, unless one manually provides a file of all possible calls. This would be too cumbersome for a large system, and is the kind of thing that one would want to have tool support for.

4.5 Other new tools – Dynamic stack usage analysis

This requires programs to be bound using the –u switch. It gives output in the format:

Index	Task Name	Stack Size	Stack usage [min - max]
1	CI Task: LAN Read	97536	[2664 - 18400]
2	CI Task: LAN Send	97536	[2568 - 18304]

The tolerance on the results is surprisingly large.

This tool is useful for checking whether any tasks are getting near their stack limit. Note though that the tasks and programs need to have been terminated cleanly, you cannot interrogate a running system to obtain a snapshot of its current usage.

4.6 Other new tools – Initialize_Scalars

This is used to look for un-initialised variables. It requires several steps:

- Add pragma Initialize_Scalars to the gnat.adc configuration file;
- Compile with the –gnatVaM switch;
- Bind with –S switch to say what value to store in otherwise un-initialised variables.

There were 35 cases found. One discovered the cause of a known problem, though most cases were fields in records where another field would indicate whether or not data was present in the first

It is a tedious method since it causes Constraint_Errors to be raised, which then have to be fixed and the system re-built, for one or two cases at a time, before one can continue testing. It has never-the-less proven very useful.

4.7 Other new tools – Warnings for un-initialised out parameters

Previously the compiler only gave a warning if an out parameter was not set anywhere. It now analyses paths and gives a warning if there is any path that does not set the out parameter.

There were 39 cases found, though in some cases there would be a second out parameter indicating whether or not data was present in the first.

4.8 Other new tools – Back-end switch for further searching for un-initialised variables

This uses the –Wuninitialized compiler switch. This largely gave false positives, e.g. in out mode parameters that only needed to be out mode, or fields that were filled in outside of Ada. It is of course wise to check all warnings even if in most cases the recommendation is “no change”.

5 Conclusions

It proved to be quite straightforward to port to Ada 2005. The Ada 95 to 2005 porting exercise required only about 10% of the effort of the preceding Ada 83 to 95 porting. Less than 1% of files were affected and the changes required were small

It was slightly harder to use call-backs and limited types, rather going against some of the design intentions of Ada 2005.

Anonymous access types and containers have already proven useful. The “not null” qualification of access types potentially will allow the earlier detection of access errors and avoid unnecessary replication of access checks.

A comparison of industrial coding rules

J-P. Rosen

Adalog, 19-21 rue du 8 mai 1945, 94110 ARCUEIL, France; email: rosen@adalog.fr

Abstract

AdaControl [1] is a (free) tool whose purpose is to enforce coding standards and programming rules in Ada programs. As AdaControl is more and more widely used in the industry, we had to review many industrial coding standards, in order to write the corresponding AdaControl rules.

This paper presents our experience with rules of various origins, analyzes the rules commonly encountered, and provides some lessons-learned about good and bad programming rules.

1 Introduction

With the raising of the use of its AdaControl tool, Adalog has developed a growing activity in consulting and services related to the checking of programming rules. This includes helping QA people to define rules, improving AdaControl to support new rules, and performing code reviews (both automatically and manually).

This activity has lead us to reviewing coding standards from many origins, but mainly from safety critical domains: air-traffic management, avionics, railway control… One could think that rules from these domains should be, more or less, the same. If there is effectively a core of generally accepted rules, there are also differences, for good and sometimes bad reasons. In this paper, we first present a classification of commonly encountered rules, then we discuss the importance of automatically checking the rules, and finally present some lessons learned.

2 Classification of rules

This "classification" is not intended as a formal taxonomy, but rather as an experimental categorization of the programming rules, intended to show the strengths, but also the difficulties and sometimes the weaknesses of many rules.

2.1 General useful rules

Some rules are of general interest, have clearly only benefits. and are therefore commonly found. For example, most projects require "only one statement/declaration per line", "no single array declarations", "unit name must be repeated after end"…

As another example, a simple and common rule is to require that every use of an identifier uses the same casing as in its declaration.

Some rules are very useful but extremely difficult to enforce by manual inspection. For example, the "no local hiding" rules forbids a local name from hiding an identical name in an outer scope; it prevents confusion of variables that depend on visibility rules.

Many projects do not use certain features of the language, like tasking or tagged types. This results in general from a design decision, made at the very beginning of the project. It is then a good practice to explicitly forbid the use of the corresponding language features.

The rule that prevents use of the `'Address` attribute is also commonly found, and is an important one, but for a special reason. Although there are very legitimate uses of addresses, experience shows that very often, use of `'Address` results from insufficient knowledge of the possibilities of Ada by people who come from other languages with insufficient training. The goal of this rule is thus not to prevent all usage of `'Address`, but to make sure that any use of it is justified and pair-reviewed.

2.2 Trivial rules

Some commonly found rules are of minimal value, simply because they are always obeyed in practice. We call these rules "trivial" because they might well be the only rules that we never found violated in any project we had to review!

For example, almost every coding standard forbids using the goto statement. Although the reasons are obvious, it is, in practice, extremely rare to find violations.

Another example is a rule that forbids declaring identifiers with the same names as entities defined in Standard. Of course, violating this rule could cause horrible confusion, but in practice, few programmers even *know* that they are allowed to declare identifiers that hide the ones from Standard!

It is also common to have a "rule" that forbids the use of TAB characters in programs. Although there are of course good reasons for it, it is hardly a rule; most editors have features to eliminate tabs, so they go away without the programmer being even aware of it. And otherwise, it is very easy to write a simple clean-up program.

2.3 Redundant rules

It is very common to find rules that repeat other rules, in a slightly different way, because they appear in a different context or were defined for a different purpose.

For example, a rule may explicitly require that, when assigning fields of records, there be only one field assignment per line. This rule is obviously redundant with the more general "one statement per line" rule. Another example is a general rule that states that "a package spec should export only entities that are used by other units", and then have a rule that states that "if a type is declared in

a package specification and used only in the body, it shall be moved to the body".

Such redundancies are annoying, because they are useless and increase artificially the number of rules. Moreover, a violation can (must?) be traced to several rules, thus making reporting more difficult.

2.4 Layout and comments rules

Some guidelines go into deep details about the number of characters that should be used for indentation, maximum length of a line and how long lines should be folded, how aggregates should be aligned, etc. A uniform presentation is an important issue as far as understandability and uniformity are concerned, however checking these rules manually is almost impossible, and writing a tool to check them automatically is roughly equivalent to writing the corresponding reformatter. It is therefore better to require the use of a reformatter (which is now included in every syntactic editor) and go with whatever layout the reformatter does, than to require a presentation that does not correspond to any tool. Uniformity is important, exact details of layout are not.

Various rules deal with comments. The easiest ones are those that require a standard header for every compilation unit. Automatic checking shows that this kind of rule is harder to enforce than one may think. Although the headers *look* conformant, there are very often small differences, like extra comment lines, missing separators, incorrect number of spaces at various places…

Header comments of subprograms are more difficult to check, since they are expected to describe the purpose of the subprogram and the semantics of the parameters – something that can be checked only manually.

Sometimes, there is a requirement that certain declarations (types, variables) be commented. Once again, a manual check is required for this kind of rule, but systematic checking requires inspecting *all* the code – something that cannot be performed routinely. There is therefore a high risk that such a rule stays as "recommended practice" without systematic checking.

Finally, some projects require a density of comments in the code (like "there must be 20% of comment lines"). In one project, the rule document failed to define how the lines are counted, which raises a number of issues: are blank lines counted? Are header comments counted?

2.5 Rules that are not coding/programming rules

Many guides include rules that are more *design* or *good-practice* rules than coding/programming rules. For example, a rule that requires that "different types shall be used to represent data from different domains". Although such rules have value, they should be kept separate from programming rules, because they cannot generally be verified automatically. Typically, they should be checked by pair-review, rather than by code inspection.

2.6 Controversial rules

Some rules are controversial, in the sense that various projects take opposite decisions., either about whether to allow some constructs, or in the way the rule should be applied. Note that this is not surprising: a life-critical project may impose rules that ensure maximum safety, even at the cost of readability and maintainability, while a less critical application may choose different trade-offs.

For example, almost every project imposes naming conventions for various elements. But some projects impose separating words in an identifier by the use of capitalization and forbid underscores (like in `LineLength`), while others prohibit that style, and require words to be separated by underscores (like in `Line_Length`). Some projects require type names to start with "`T_`", or end with "`_Type`". Renamings are an interesting issue, as far as naming convention is concerned: should renamings have their own naming convention to show that they are aliases, or should they follow the rule for the renamed entity?

Using the use clause is another controversial issue: some projects disallow it altogether, other allow it only if restricted to the innermost scope where it is useful, and some place no restriction to it.

Some rules require systematic initialization of all variables at the point of declaration. Although it may seem useful to make sure that every variable receives a proper value before being used, this is an interesting case of a rule that may have adverse effects. The rule may induce people into assigning a "default" value to variables (that may not be appropriate) just to pass the check; this may in turn result in more subtle bugs than those caused by a plain non-initialized variable. For this reason, some rules forbid systematic initialization (especially when the initialization value is known to be overridden later on).

2.7 Insufficient rules

Some rules are intended for a certain purpose, but if they are not properly formulated and/or explained, they can fail to achieve their intended goal. For example, it is common to disallow the use of predefined numeric types. This is intended to promote the definition of higher level, more abstract numeric types. However, in a project, this resulted in the definition of types like "Int_8", "Int_16", and "Int_32" that were used everywhere. There was some benefit to it, as it made the program independent of the size of the predefined integer types, but did not bring the benefits expected from strong typing of numeric values.

Often, the rule does not assert all the consequences. For example, there can be a rule that says "no package shall be declared in a procedure". Such a rule is generally intended to limit the complexity of subprograms, but does it also apply to instantiations of generic packages? They are formally local packages, but the rule would prevent, for example, instantiating Integer_IO inside of an IO routine – a very legitimate construct actually.

Sometimes, rules are written with a very narrow perspective. We encountered a rule that said that "when an

array is assigned in full, all components of the aggregates should be named". But of course, assigning an array in full does not necessarily use an aggregate; and what about aggregates that appear in a context other than as the rigth hand side of an assignment? Should the rule apply to record aggregates? Clearly, the person who wrote the rule had used aggregates only in very limited contexts, and wrote the rule according to that usage.

2.8 Inappropriate rules

Sometimes, rules are clearly a legacy from other languages, or simply show ignorance about Ada. For example, a project required an order for declarations: constants, then types, then variables (and failed to define an order for Ada entities that had no Pascal equivalent, like packages and exceptions!). This was clearly a remaining from the Pascal philosophy, but prevented for example the grouping of declarations that were logically related.

In another case, a rule required the presence of an "else" part for every "if", leading to many "else null;" in the program. This rule was derived from Misra-C, where it is intended to prevent the "dangling else" problem in C. The Ada syntax (which requires "end if") does not have this problem, but the rule was reconducted anyway.

Another (funny) example is "rules" that forbid constructs that are actually not legal Ada; we have encountered a project that banned the use of anonymous array types as record components, or default initialization of array components … Such rules are harmless by themselves, but create suspicion about the validity of other rules.

A special kind of dangerous rules are those that are justified by efficiency considerations. Rules sometimes require or forbid the use of some constructs for efficiency reasons. Although this may seem justified in time-constrained software, experience shows that actual measures of the run-time cost of such structures have only very rarely been performed; often, the rule just expresses the "intimate belief" of those who wrote the rules, without the backing of hard figures. Very often, these rules are not justified at all, and may even force using less efficient constructs. Even when such rules are justified, it must be remembered that "inefficient" constructs may become very efficient with the next version of the compiler.

A special (and even worse) case of the above is rules that are intended to work around compiler bugs. Such rules tend to stay forever, years after the bug has been fixed…

Note that it is often the *motivation* of the rule which is wrong, not the rule by itself. For example, a project required short circuit forms (and then and or else) rather than plain and or or, on the ground that they were more efficient. Such a general statement is highly likely to be plain wrong – at least in some cases, and the gain in micro-efficiency does not justify the rule. On the other hand, another project had the same rule, but on the ground that it would simplify unit testing, because each logical operation would require only three tests instead of four with the regular operators. This reason was perfectly acceptable.

2.9 Good rules that are harder to enforce than they seem

Some rules are apparently well motivated, but very hard to apply in practice, or (almost) impossible to check. For example, several projects wanted to prevent the use of "magic numbers", i.e. numerical values that appear directly in the program text; instead, every such value should be given a name, as a constant or named number. Obviously, this rule cannot apply to literals used precisely in the definition of constants and named numbers. But there are many other cases where numeric literals cannot be avoided, like in representation clauses for example. And in X**2, it would be stupid to forbid the use of "2"… If taken too literally, this rule would force people to declare constants like Number_2, which would bring no benefit at all.

It is also common to find rules that prevent assignment to fields of records, in favour of whole assignments with aggregates. This is an important rule for maintainability, since the addition of a component to a record will result in illegal code everywhere the corresponding modification has been omitted. But sometimes, you just want to assign a value to one component: should you force a full aggregate assignment in this case? Let us assume for a start that an aggregate is required if every component is changed, and that single assignment to a component is allowed if no other component is changed. Where should the limit when aggregate assignment is required be placed? If more than XX components are changed? If less than YY components are not changed? If more than ZZ% of the components are affected? Making a rule which achieves the desired goal and is still practical is far from obvious.

2.10 Rules not checkable by nature

Finally, some rules are, by nature, impossible to enforce automatically, generally because they involve some value judgement. This includes rules like "parentheses should be used to improve readability", "elements should be grouped in a package according to the logical structure", and of course "identifiers should have meaningful names".

The checking of this kind of rule must be done manually. In some cases, a tool can be of help by identifying automatically the constructs that must be reviewed manually; in other cases, checking the rule requires a detailed reading of the whole source.

Actually, this kind of "rule" should really be guidelines, and separated from the true coding rules.

3 The value of a tool for checking rules

In the previous chapter, we repeatedly addressed the issue of the checkability of the rules. It is nice to issue rules, but a rule is meant to be enforced; counting on programmers' discipline simply does not work.

It must therefore be stressed that rules are of little value, unless there is a tool to enforce them. No manual inspection can approach the level of scrutiny provided by a tool; actually, *all* of our clients were greatly surprised when we ran AdaControl on their carefully reviewed code,

sometimes finding *thousands* of violations that had escaped manual inspection.

Moreover, manual inspection is a lengthy and costly process. It can be performed once for every major release of the product, for example at the time of formal certification for safety-critical software[3], but can certainly not be done routinely.

There are several such tools on the market: in addition to Adalog's AdaControl, popular tools include AdaCore's Gnatcheck, GrammaTech's Ada-Assured, LDRA's Testbed, Logiscope's Rulechecker, and RainCode's Adarc. Moreover, many compilers include options to enforce coding rules at compile time. Some rules can even be enforced by the language with the use of **pragma** restriction.

An important issue when choosing a tool is ease of use in day-to-day development. When rules checking is performed late in the development process, one discovers generally a huge amount of violations, and fixing them requires a tremendous effort; it is sometimes extremely difficult to do when the software has already gone through various validation phases that would be ruined by massive corrections. When the tool is integrated into the development environment, programmers can run it routinely each time they develop new modules or modify existing ones, ideally by simply clicking a button in their favourite IDE. The sooner checking is performed in the development process, the better.

From this point of view, it could seem useful to have rules checked directly by the compiler. But compilers do not have such sophisticated and parameterizable rules like dedicated tools have. Unlike language rules, programming rules depend heavily on the kind and constraints of the project; parameterization is therefore absolutely necessary. Moreover, rules checking must also be performed by quality assurance people, at the time of integration. Having some rules checked by the compiler while other still require the use of another tool would force QA people to run two tools as part of the process, with different outputs that are hard to merge. Therefore, even if the compiler does some checks, it is important that the rule checking tool be able to enforce also rules checked by the compiler.

4 Lessons learned

4.1 How to define "good rules"

Providing a good set of programming rules is not easy. Sometimes, it seems that rules are there just for the sake of having rules; occasionally, rules may have an effect opposite to their intent.

It is therefore important that every rule be motivated and justified. Some of the questions that need be answered to check the value of a rule are:

- What is the problem that this rule will prevent/minimize?
- Is this rule really necessary?
- What are the possible adverse or perverse effects of the rule?
- Is this rule automatically checkable?
- What are the cases where the rule should *not* be obeyed?

Of course, it does not make sense to reinvent the wheel every time. A programming rules document should start from some existing and recognized document, like the famous "Ada Quality and Style Guide"[2], which is actually a generic template intended precisely to serve as the basis for coding standards. It was surprising that, among the documents we reviewed, many of them didn't even quote the Ada Q&S Guide, although they often referred to coding standards from other languages… Another valuable source of inspiration is the NASA coding standard for the Goddard Dynamic Simulator, which is freely available on the internet [3].

Coding rules should really be coding rules. They should be defined separately from design rules, and also from guidelines, which are common sense recommendations that cannot be specified – and even less checked – formally.

Rules should be proposed by QA people, but should be reviewed and discussed with programmers and language experts. Otherwise, there is a risk that the cost of a rule, even a perfectly reasonable one, be higher than its benefits, for reasons linked to the technical details of the project.

It should be also understood that developing a good set of rules is an iterative process; experience shows that some rules are useless, some have an adverse effect, and some are missing. There should be a process for getting feedback from the developers and improving the rules document.

4.2 Derogations

When a rule is proposed, it is very important to be aware that there *will* be cases where the rule should *not* be obeyed. Derogations to a rule are normal; however, derogations should only be granted by QA, after review and justification.

Failing to recognize the need for derogations can lead to two equally bad effects:

- Either force application of the rule in any case, often resulting in twisting the code to match the rule with a very poor result as far as quality is concerned
- Or simply abandon the rule, on the ground that it cannot *always* be applied.

Therefore, every coding standard should include a process for requesting a derogation, and tools should provide a way to ignore violations at indicated places. The process for granting a derogation when appropriate should not be too heavy; otherwise, it may appear simpler to the programmer

[3] But at that time, it is generally too late to correct massive violations, and the project ends up with a document to justify why the violations are not safety-critical, rather than fixing them.

to obey by the rule, even where not appropriate, rather than to request a derogation.

4.3 Form of the document

The coding rules document should ideally specify, for each rule:

- The statement of the rule
- The motivation for the rule
- An example where the rule is obeyed
- An example where the rule is not obeyed
- Cases where the rule is *not* applicable
- Whether and how the rule can be checked by automatic tools

The goal of this is to make sure that the programmers understand the rule, understand and accept the motivation of the rule, know how to check it, and know how to ask for a justified derogation.

Since such a document can become rapidly quite thick, having a quick summary of the rules with pointers to the full explanation can make the document much more usable.

4.4 Communication

Coding standard should be perceived by programmers as a help rather than a burden. It is of course important to have clear and easily accessible documents to describe the rules, but organizing team meetings, where the rules are presented and their motivations explained, can be very effective. Such general presentations bring several benefit:

- they provide feed-back from the base to the QA people, often resulting in improvements to the rules;
- they make acceptance of the rules easier; people have no problem following rules when they understand their purpose;
- with sufficient tool support, it will help making the checking of the rule a routine, therefore catching violations early in the development process and avoiding massive rewritings.

5 Conclusion

A good set of programming rules is one which really contributes to the quality of the code without putting unnecessary burden on the programmer, is precisely defined and well understood by all users, and easily enforceable by automated tools. Defining such a set of is far from easy: some rules are general, but others depend on the particular context of the application.

It must be acknowledged that programming rules have to be refined iteratively, and that good communications between QA people and users is a key to achieving a set of rules that really improves the quality of the project.

References

[1] http://www.adalog.fr/adacontrol2.htm

[2] Software Productivity Consortium, "Ada 95 Quality and Style Guide".

[3] Stephen Leake, "Goddard Dynamic Simulator, Ada Coding Standard", http://fsw.gsfc.nasa.gov/gds/code_standards_ada.pdf.

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 #23: Null Considered Harmful

Bob Duff, AdaCore

Date: 14 January 2008

Abstract: The "not null" syntax allows an Ada 2005 program to prevent access values from being null in cases where the null value is undesirable. This new syntax helps provide useful documentation.

Let's get started…

Ada, like many languages, defines a special 'null' value for access types. All values of an access type designate some object of the designated type, except for null, which does not designate any object. The null value can be used as a special flag. For example, a singly-linked list can be null-terminated. A Lookup function can return null to mean "not found", presuming the result is of an access type:

```
type Ref_Element is access all Element;
Not_Found : constant Ref_Element := null;
function Lookup (T : Table) return Ref_Element;
- -  Returns Not_Found if not found.
```

An alternative design for Lookup would be to raise an exception:

```
Not_Found : exception;
function Lookup (T : Table) return Ref_Element;
- - Raises Not_Found if not found.
- - Never returns null   <–  Ada 95 comment.
```

Neither design is better in all situations; it depends in part on whether we consider the "not found" situation to be exceptional.

Clearly, the client calling Lookup needs to know whether it can return null, and if so, what that means. In general, it's a good idea to document whether things can be null or not, especially for formal parameters and function results. In Ada 95, we do that with comments. In Ada 2005, we can use the "not null" syntax:

```
function Lookup (T : Table)
  return not null Ref_Element;  - - Ada 2005
```

In general, it's better to use the language proper for documentation, when possible, rather than comments, because compile-time and/or run-time checks can help ensure that the "documentation" is actually true. With comments, there's a greater danger that the comment will become false during maintenance, and false documentation is obviously a menace.

In many, perhaps most, cases, null is just a tripping hazard. It's a good idea to put in "not null" when possible. In fact, a good argument can be made that "not null" should be the default, with extra syntax required when null is wanted. This is the way SML works, for example — you don't get any special null-like value unless you ask for it. Of course, Ada 2005 needs to be compatible with Ada 95, so "not null" cannot be the default for Ada.

One word of caution: access objects are default-initialized to null, so if you have a "not null" object (or component) you had better initialize it explicitly, or you will get Constraint_Error. "Not null" is more often useful on parameters and function results, for this reason.

Gem #24: Null Considered Harmful (Part 2 — Efficiency)

Bob Duff, AdaCore

Date: 28 January 2008

Abstract: The "not null" syntax can make programs more efficient by removing the need for implicit run-time checks.

Let's get started…

In last week's gem, we talked about the documentation advantages of using "not null". Here's another example, first with null:

```
procedure Iterate
  (T : Table;
   Action : access procedure (
                        X : not null Ref_Element)
     := null);
 - -  If Action is null, do nothing.
```

…and without null:

```
procedure Do_Nothing (X : not null Ref_Element)
   is null;
procedure Iterate
  (T : Table;
   Action : not null access procedure (
                        X : not null Ref_Element)
     := Do_Nothing'Access);
```

I much prefer the style of the second Iterate.

The "not null access procedure" is quite a mouthful, but it's worthwhile, and anyway, the compatibility requirement for Ada 2005 requires that the "not null" be explicit, rather than the other way around.

Another advantage of "not null" over comments is for efficiency. For example:

```
procedure P (X : not null Ref_Element) is
begin
   X.all.Component := X.all.Component + 1;
end P;

procedure Q (X : not null Ref_Element) is
```

```
begin
   while … loop
      P (X);
   end loop;
end Q;

procedure R is
begin
   Q (An_Element'Access);
end R;
```

Without "not null", the generated code for P will do a check that X /= null, which may be costly on some systems. P is called in a loop, so this check will likely occur many times. With "not null", the check is pushed to the call site. Pushing checks to the call site is usually beneficial because (1) the check might be hoisted out of a loop by the optimizer, or (2) the check might be eliminated altogether, as in the example above, where the compiler knows that An_Element'Access cannot be null.

This is analogous to the situation in Ada 95 with other run-time checks, such as array bounds checks:

```
type My_Index is range 1..10;
type My_Array is array (My_Index) of Integer;
procedure Process_Array (X : in out My_Array;
                         Index : My_Index);
```

If "X (Index)" occurs inside Process_Array, there is no need to check that Index is in range, because the check is pushed to the caller.

Gem #33: Accessibility Checks (Part I: Ada95)

Ramón Fernández-Marina, AdaCore

Date: 28 April 2008

Abstract: The existence of dangling references (pointers to objects that no longer exist) in a program can have catastrophic results. Ada incorporates a set of "accessibility rules" that help the programmer prevent dangling references, making programs more secure.

Let's get started…

Ada is a block-structured language, which means the programmer can nest blocks of code inside other blocks. At the end of a block, all objects declared inside of it go out of scope, meaning they no longer exist, so the language disallows pointers to objects in blocks with a deeper nesting level.

In order to prevent dangling references, every entity is associated with a number, called its "accessibility level", according to a Ada's accessibility rules. When certain references are made to an entity of an access type (Ada's parlance for pointer), the accessibility level of the entity is checked against the level allowed by the context so that no dangling pointers can occur.

Consider the following example:

```
procedure Static_Check is
   type Global is access all Integer;
   X : Global;

   procedure Init is
      Y : aliased Integer := 0;
   begin
      X := Y'Access; - - Illegal!
   end Init;

begin
   Init;
   …
end Static_Check;
```

The assignment is illegal because when the procedure Init finishes, the object Y no longer exists, thus making X a danging pointer. The compiler will detect this situation and flag the error.

The beauty of the accessibility rules is that most of them can be checked and enforced at compile time, just by using statically known accessibility levels.

However, there are cases when it is not possible to statically determine the accessibility level that an entity will have during program execution. In these cases, the compiler will insert a run-time check to raise an exception if a dangling pointer can be created:

```
procedure Access_Params is
   type Integer_Access is access all Integer;
   Data : Integer_Access;

   procedure Init_Data (Value : access Integer) is
   begin
      Data := Integer_Access (Value);
      - - this conversion performs a dynamic
      - - accessibility check
   end;

   X : aliased Integer := 1;

begin
   Init_Data (X'Access); - - This is OK

   declare
      Y : aliased Integer := 2;
   begin
      Init_Data (Y'Access); - -  Trouble!
   end;
    - -  Y no longer exists!

    Process (Data);
end;
```

In the example above, we cannot know at compile time the accessibility level of the object that will be passed to Init_Data, so the compiler inserts a run-time check to make sure that the assignment 'Data := …' does not cause a dangling reference — and to raise an exception if it would.

In summary, when it comes to dangling references, Ada makes it very hard for you to shoot yourself in the foot!

Gem #41: Accessibility Checks (Part II: Ada2005)

Ramón Fernández-Marina, AdaCore

Date: 30 June 2008

Abstract: Gem #41 — Ada 2005 brings a number of improvements concerning access types, aimed at simplifying the programmer's task and adding flexibility to the language. But with greater power comes greater responsibility, so accessibility checks have also been extended to prevent these new features from creating dangling pointers.

Let's get started…

Ada 2005 allows the use of anonymous access types in a more general manner, adding considerable power to the object-oriented programming features of the language. The accessibility rules have been correspondingly augmented to ensure safety by preventing the possibility of dangling references. The new rules have been designed with programming flexibility in mind, as well as to allow the compiler to enforce checks statically.

The accessibility levels in the new contexts for anonymous access types are generally determined by the scope where they are declared. This makes it possible to perform compile-time accessibility checks.

Another rule that allows for static accessibility checks relates to derived types: a type derivation does not create new accessibility level for the derived type, but just takes that of the parent type:

```
procedure Example_1 is
   type Node is record
      N : access Integer;
   end record;
   List : Node

   procedure P is
      type Other_Node is new Node;
   begin
      declare
         L : aliased Integer := 1;
         Data : Other_Node :=
              Other_Node'(N => L'Access);
         - -  L'Access is illegal!
      begin
         List := Node (Data);
      end;
   end P;

begin
   P;
end Example_1;
```

In the above example, we don't need to worry about expensive run-time checks on assignment or return of an object of type Other_Node; we know it has the same accessibility level as type Node, making the Access attribute illegal. If this were not prevented, after returning from P, List.N would be a dangling reference.

Ada 2005 also allows functions to return objects of anonymous access types. In this case, the accessibility level of the object is statically determined by the scope of the function declaration. Consider the following example:

```
procedure Example_2 is
   type Rec is record
      V : access Integer;
      …
   end record;

   Global : aliased Integer := 1;

   function F1 (X : Boolean) return Rec is
      Local : aliased Integer := 2;

      - -  Nested function returns anonymous
      - - access values
      - -  with different nesting depths

      function F2 (Y : Boolean)
         return access Integer is
      begin
         if Y then
            return Global'Access;
         else
            return Local'Access;
         end if;
      end F2;

   begin
      return (V => F2 (X), …); - - Illegal
   end F1;

   Data : Rec;
begin
   Data := F1 (True);
end Example_2;
```

In this example, applying the aforementioned rule, the compiler statically determines that this accessibility level is the scope where F2 is declared, which is deeper than the accessibility level of Rec. So even though the call F1 (True) would provide a valid value for V, the code is illegal. The accessibility restriction is conservative, to keep the rules simple, and so that the compiler is not required to perform data flow analysis to determine legality (not to mention that in general the legality would be undecidable).

The new rules also take into account discriminants of an anonymous access type (which are technically referred to as access discriminants). In Ada 2005, access discriminants are now permitted for nonlimited types. Consequently, it's necessary to disallow defaults for access discriminants of nonlimited types. Thus, the following declaration is illegal:

```
Default : aliased Integer := …
type Rec (D : access Integer := Default'Access)
   is record
   …
```

This restriction is needed to prevent the discriminant from creating a dangling reference due to an assignment of the record object; it ensures that the object and the discriminant are bound together for their lifetime.

Special care must be taken when types with access discriminants are used with allocators and return statements. The accessibility rules require the compiler to perform static checks when new objects containing access discriminatns are created or returned. Consider the following example:

```
procedure Example_3 is
   type Node (D : access Integer) is record
      V : Integer;
   end record;
   type Ptr is access all Node;

   Global_Value : aliased Integer := 1;
   Other_Data   : Integer := 2;

   procedure P is
      Local : aliased Integer := 3;
      R1 : Ptr;
      R2 : Ptr;
   begin
      R1 := new Node'(D => Global_Value'Access,
                      V => Other_Data);
      - -  This is legal

      R2 := new Node'(D => Local_Value'Access,
                      V => Other_Data);
      - -  This is illegal
   end P;
begin
   null;
end Example_3;
```

The allocator for R1 is legal, since the accessibility level of Global'Access is the same as the accessibility level of D. However the allocator for R2 is illegal, because the accessibility level of Local'Access is deeper than the accessibility level of D, and assigning R2 to an object outside P could lead to a dangling reference.

In summary, these rules forbid the creation of an object in a storage pool that contains an access discriminant pointing to some area of memory, be it a part of the stack or some other storage pool, with a shorter lifetime, thus preventing the discriminant from pointing to a nonexistent object.

Gem #44: Accessibility Checks (Part III)

Bob Duff, AdaCore

Date: 15 September 2008

Abstract: 'Unchecked_Access can be used to bypass the accessibility rules, and controlled types can be used to rein in this dangerous feature.

Let's get started…

In Parts #1 and #2, we showed how the accessibility rules help prevent dangling pointers, by ensuring that pointers cannot point from longer-lived scopes to shorter-lived ones. But what if you want to do that?

In some cases, it is necessary to store a reference to a local object in a global data structure. You can do that by using 'Unchecked_Access instead of 'Access. The "Unchecked" in the name reminds you that you are bypassing the normal accessibility rules. To prevent dangling pointers, you need to remove the pointer from the global data structure before leaving the scope of the object.

As for any unsafe feature, it is a good idea to encapsulate 'Unchecked_Access, rather than scattering it all around the program. You can do this using limited controlled types. The idea is that Initialize plants a pointer to the object in some global data structure, and Finalize removes the pointer just before it becomes a dangling pointer.

Here is an example. Let's assume there are no tasks, and no heap-allocated objects — otherwise, we would need a more complicated data structure, such as a doubly-linked list, with locking. We keep a stack of objects, implemented as a linked list via Stack_Top and chained through the Prev component. All occurrences of 'Unchecked_Access are encapsulated in the Objects package, and clients of Objects (such as Main, below at end) can freely declare Objects, without worrying about dangling pointers. Stack_Top can never dangle, because Finalize cleans up, even in the case of exceptions and aborts.

Note that 'Unchecked_Access is applied to a formal parameter of type Object, which is legal because formals of tagged types are defined to be aliased. Note also that Print_All_Objects has no visibility on the objects it is printing.

This program prints:

```
Inside Nested:
  That_Object
  This_Object
After Nested returns:
  This_Object
```

Observe that That_Object is not printed by the second call to Print_All_Objects, because it no longer exists at that time.

```
private with Ada.Finalization;
package Objects is

   type Object (Name : access constant String)
     is limited private;
   - -  The Name is just to illustrate what's going on
   - -  by printing it out.

   procedure For_All_Objects (
      Action : not null access procedure (X : Object));
   - -  Iterate through all existing Objects in
   - -  reverse order of creation,
   - -  calling Action for each one.

   procedure Print_All_Objects;
   - -  Print out the Names of all Objects in
   - -  reverse order of creation.

   - -  … other operations

private
   use Ada;

   type Object (Name : access constant String)
      is new Finalization.Limited_Controlled with
     record
```

```
      - -  … other components
      Prev : access Object
                  := null; - - previous Object on the stack
   end record;

   procedure Initialize (X : in out Object);
   procedure Finalize (X : in out Object);

end Objects;

with Ada.Text_IO;
package body Objects is

   Stack_Top : access Object := null;

   procedure Initialize (X : in out Object) is
   begin
      - -  Push X onto the stack:
      X.Prev := Stack_Top;
      Stack_Top := X'Unchecked_Access;
   end Initialize;

   procedure Finalize (X : in out Object) is
   begin
      pragma Assert (Stack_Top = X'Unchecked_Access);
      --  Pop X from the stack:
      Stack_Top := X.Prev;
      X.Prev := null;  - -  not really necessary, but safe
   end Finalize;

   procedure For_All_Objects (
      Action : not null access procedure (X : Object)) is
      - -  Loop through the stack from top to bottom.
      Item : access Object := Stack_Top;
   begin
      while Item /= null loop
         Action (Item.all);
         Item := Item.Prev;
      end loop;
   end For_All_Objects;

   procedure Print_All_Objects is
      - -  Iterate through the stack using
      - - For_All_Objects, passing
      - -  Print_One_Object to print each one.
      procedure Print_One_Object(X : Object) is
      begin
         Text_IO.Put_Line ("   " & X.Name.all);
      end Print_One_Object;
   begin
      For_All_Objects (Print_One_Object'Access);
   end Print_All_Objects;

end Objects;

with Ada.Text_IO; use Ada;
with Objects; use Objects;
procedure Main is

   This_Object : Object (
      Name => new String'("This_Object"));

   procedure Nested is
      That_Object : Object (
           Name => new String'("That_Object"));
   begin
      Text_IO.Put_Line ("Inside Nested:");
      Print_All_Objects;
   end Nested;

begin
   Nested;
   Text_IO.Put_Line ("After Nested returns:");
   Print_All_Objects;
end Main;
```

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o K.U. Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Peter Dencker
Steinäckerstr. 25
D-76275 Ettlingen-Spessartt
Germany
Email: dencker@web.de
URL: ada-deutschland.de

Ada-France

Association Ada-France
c/o Jérôme Hugues
Département Informatique et Réseau
École Nationale Supérieure des Télécomunications
46, rue Barrault
75634 Paris Cedex 135
France
Email: bureau@ada-france.org
URL: www.ada-france.org

Ada-Spain

attn. José Javier Gutiérrez
Ada-Spain
P.O.Box 50.403
28080-Madrid
Spain
Phone: +34-942-201-394
Fax: +34-942-201-402
Email: gutierjj@unican.es
URL: www.adaspain.org

Ada in Sweden

attn. Rei Stråhle
Saab Systems
S:t Olofsgatan 9A
SE-753 21 Uppsala
Sweden
Phone: +46 73 437 7124
Fax: +46 85 808 7260
Email: Rei.Strahle@saabgroup.com
URL: www.ada-i-sverige.se

Ada 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