ADA USER JOURNAL

Volume 29
Number 1
March 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Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.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

In this first editorial of 2008, I would like to take a brief moment to comment on some highlights the Ada Europe community can expect this year. First, I am pleased that before long Ada-Europe members will receive another package, containing the Rationale for Ada 2005, written by John Barnes, as a volume of the Springer LNCS series. This book, produced by Ada-Europe, provides a good opportunity for all of us to (re)visit the changes and additions Ada 2005 brings us.

The year will also witness another edition of our Ada-Europe conference, which takes place in the 3rd week of June in the beautiful scenery of Venice, Italy. You will find more information concerning this worthwhile event in the Forthcoming Events section of the Journal, together with the call for contributions for the SIGAda conference next October in Portland, USA.

Continuing with the contents of this issue, the first paper is a contribution of David Kleidermacher of Green Hills Software, USA, discussing the practical use of static analysis in embedded systems. This is the final paper coming from the Industrial Track of the Ada-Europe 2007 conference, a forum that has, and I believe will continue to, provided valuable content to the Journal. I am looking forward for the papers from the Industrial Track of Ada-Europe 2008, as well as from the conference's "Ada and Education" session, which the Journal will also host.

The issue also continues with the Proceedings of the 13th International Real-Time Ada Workshop (IRTAW-13), providing the contents of the second session of the workshop: Programming Languages and Patterns. As usual, the first paper provides the session report. The first two technical papers of the session come from the University of York, UK. In the first, the authors argue in favour of a standardised library of real-time utilities for Ada 2005, whilst in the second the authors elaborate on specific utilities for implementing execution-time servers. The session closes with a paper from a group of authors coming from the Technical University of Madrid, Spain, Télécom Paris, France and University of Padua, Italy, presenting a set of coding patterns to support automated code generation of meta-models for high-integrity systems.

Continuing with the contributions from the Gem of the Week series, this issue provides Matthew Heaney's Gems on the topic of Containers. And, as usual, you will find the valuable information of the News and Calendar sections, contributed by Santiago Urueña and Dirk Craeynest, their respective editors.

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

News

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

Contents

Ada-related Organizations

ARA — Ada 2005 Rationale update

From: Ada Resource Association
Date: January 4, 2008
Subject: Ada 2005 Rationale
URL: http://adaic.com/whatsnew.html

The Ada 2005 Rationale has been updated, fixing a number of errors, enhancing the index, and adding a new Postscript.

[See also "ARA — Ada 2005 Rationale Available" in AUJ 27-1 (Mar 2006) —su]

ARA — ACATS 3.0

From: Ada Resource Association
Date: January 25, 2008
Subject: Ada Conformity Assessment Test Suite
URL: http://adaic.com/whatsnew.html

The first Ada 2005 Ada Conformity Assessment Test Suite, ACATS 3.0, has been posted, along with an associated ACATS Modification List 3.0A.

Update: ACATS Modification List 3.0B and the associated test files have been posted.

[See also "ARA — Development snapshot for ACATS 3.0" in AUJ 28-2 (Jun 2007) —su]

Ada-related Events

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

Ada UK videos online

From: AdaCore Developer Center
Date: Wednesday February 13, 2008
Subject: Ada UK videos online
RSS: http://www.adacore.com/2008/02/13/ada-uk-videos-online/

The videos of the presentations given at the recent Ada event in the UK can now be viewed here:

http://www.adacore.com/home/ada_answers/lectures/ada_uk07

Topics include:

- Sufficient Evidence?
- Porting to Ada 2005
- Can Ada be used with Multiple Independent Levels of Security?
- The Marte run-time and the advantages Ada has for real-time programmers
- Correctness by Construction: Putting Engineering into Software
- The automatic extraction of semantic information using advanced static analysis
- The DO-178C standardization process and implications for language
- Using Ada for software development tools
- The future of programming languages

Dec 6 — Ada-France 2007

From: Ada-France
Subject: Présentations faites lors de la journée Ada 6 décembre 2007
Date: Monday December 10, 2007
URL: http://www.ada-france.org/article138.html

[Translated from French. —su]

Ada-France has organised December 6, 2007, a technical seminar about embedded systems at Telecom Bretagne. This seminar brought together about thirty participants.

The presentations were:

- Premiers retours d'un chercheur sur l'utilisation de l'IDM pour le temps réel. Jérôme Delatour, ESEO (Angers).
- AADL: état et perspectives. Pierre Dissaux, Ellidiss Technologies (Brest) .
- Validation de systèmes temps-réel et embarqué à partir d'un modèle MARTE : expérimentation. Eric Maes, Thales Research and Technology (Palaiseau) .
- AUTOSAR: Streamlining automotive systems and processes. Francois Dupont, Geensys (Brest) .
- Expérimentation d'unités de preuve pour la validation formelle de logiciels embarqués critiques. Philippe Dhaussy*, Pierre Yves Pilain*, Dominique Kerjean*, Stéphane de Belloy**, Arnaud Monégier du Sorbier**, Hugues Bonnin +, Frédéric Boniol***. * Laboratoire DTN, NSIETA (Brest), ** Thales AIR SYSTEMS (Rungis), + CS-SI (Toulouse), *** IRIT-ENSEEIHT (Toulouse).
- Ada 2005 pour les systèmes embarqués temps réel. José F. Ruiz, AdaCore (Paris).
- Les outils de retro-ingénierie de code Ada. Eric Audrezet, Sodius (Nantes).

[See also same topic in AUJ 28-4 (Dec 2007) —su]

Feb 24 — Ada Deutschland

From: AdaCore Press Center
Date: Thursday December 20, 2007
Subject: Efficient Development of Reliable Software Workshop
RSS: http://www.adacore.com/2007/12/20/efficient-development-of-reliable-software-workshop/

Ada Deutschland has organized a one day event round the topic of:

Efficient Development of Reliable Software and Related Methods.

Jóse Ruiz will be giving a talk on "Ada 2005 for real-time, embedded and high-integrity systems".

June 16–20 — Ada-Europe 2008

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Date: Sun, 6 Jan 2008 22:03:27 +0100 (CET)
Organization: Ada-Europe, c/o Dept. of Computer Science, K.U.Leuven
Subject: FINAL CfIP, Conference Reliable Software Technologies, Ada-Europe 2008
Newsgroups: comp.lang.ada, fr.comp.lang.ada,comp.lang.misc

Summary: One week until submission deadline!

Keywords: Conference,tutorials,industry, reliability, Ada,LNCS,Venice,Italy

FINAL Call for Industrial Presentations

13th International Conference on Reliable Software Technologies — Ada-Europe 2008
16 – 20 June 2008, Venice, Italy
http://www.ada-europe.org/conference2008.html

The 13th International Conference on Reliable Software Technologies (Ada-

Europe 2008) will take place in Venice, Italy. 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.

In addition to the usual call for papers, the conference also seeks industrial presentations which may have 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 (tullio.vardanega@math.unipd.it) by 13 January 2008. 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 2008, 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.

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

Schedule

13 January 2008: Submission of industrial presentation proposals

03 February 2008: Notification to all authors

11 May 2008: Industrial presentation material required

16-20 June 2008: Conference

Industrial Committee Members (preliminary list)

Guillem Bernat, Rapita Systems

Olivier Devuns, Aonix

Franco Gasperoni, AdaCore

Rei Stråhle, Saab Systems

Tullio Vardanega, Ada-Europe (President)

Dirk Craeynest, Ada-Europe (Vice-President)

Oct 26–30 — SIGAda 2008

From: Michael Feldman <mfeldman@seas.gwu.edu>
Newsgroups: comp.lang.ada
Subject: Call for contributions to SIGAda 2008, Portland Oct. 26-30
Date: Tue, 22 Jan 2008 20:41:33 −0000
Organization: The George Washington University

Call for Technical Contributions — SIGAda 2008

ACM SIGAda Annual International Conference

Toward Safe, Secure, Reliable Software

October 26–30, 2008

University Place Hotel and Conference Center

Portland, Oregon, USA

Sponsored by ACM SIGAda, the ACM Special Interest Group on Ada

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

(ACM Approval Pending)

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.

CONFERENCE LOCATION

Portland is the attractive, livable "City of Roses" in the Pacific Northwest. The weather in October is usually cool and often beautiful. University Place is a modern and reasonably-priced hotel located within walking distance of the central business district, the lively riverfront area, and the Portland State University campus.

HOW YOU CAN CONTRIBUTE

SIGAda 2008 solicits contributions in six major categories: Technical Articles, Extended Abstracts, Experience Reports, Workshops, Panel Sessions, and Tutorials.

Contributions from students and faculty are actively solicited, as are experience reports from practitioners.

Final acceptance will be contingent on at least one co-author registering for and presenting the contribution at the Conference.

POSSIBLE TOPICS include but are not limited to:

- Transitioning to Ada 2005
- Educational challenges for developing reliable, safe, secure software
- Ada and SPARK in the classroom and student laboratory
- Language selection for a high reliability system
- Use of high reliability subsets or profiles such as MISRA C, Ravenscar, SPARK
- High reliability standards and their issues
- Software process and quality metrics
- Analysis, testing, and validation
- Use of ASIS for new Ada tool development
- Mixed-language development
- High-reliability development experience reports
- Static analysis of code
- Integrating COTS software components
- System Architecture & Design
- Information Assurance
- Ada products certified against Common Criteria / Common Evaluation Methodology

TECHNICAL ARTICLES present significant results in research, practice, or education. These papers will be double-blind refereed and published in the Conference Proceedings and in Ada Letters.

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. Clearly state the contribution of the work being described, 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 to the design and implementation of applications such as the following: avionics, aerospace, automobile, command and control, consumer electronics, process control, transportation, trading systems, energy, medical systems, simulation, telecommunications, etc. Such reports will be selected on the basis of the interest of the experience presented to the community of Ada practitioners. 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 SIGAda 2008 Advance Program. Workshop proposals will be evaluated by the Program Committee and selected based on their applicability to the conference and potential for attracting participants. Proposals should state the problem or issue to be addressed, the coordinator(s), and criteria for participant selection.

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.

SUBMISSION DEADLINE: May 12, 2008

HOW TO SUBMIT:

Send contributions in Word, PDF, or text format as follows:

Technical Articles, Extended Abstracts, Experience Reports, and Panel Session

Proposals: Program Chair, Leemon C. Baird III (leemon.baird at usafa.edu).

Workshop proposals: Workshops Chair, Bill Thomas (BThomas at MITRE.org).

Tutorial proposals: Tutorials Chair, David A. Cook (DCook at AEgisTG.Com).

- 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.

- VENDORS. Please contact S. Ron Oliver (SROliver at CSC.CalPoly.Edu) for information about participation at SIGAda 2008.

Please submit any questions on the conference to the Conference Chair, Michael Feldman (mfeldman at gwu.edu).

IMPORTANT VISA INFORMATION FOR NON-US SUBMITTERS

General Visa Information

The sites http://www.UnitedStatesVisas.gov and http://travel.state.gov have information about obtaining a visa for those traveling to the United States. Both sites have links to websites for U.S. embassies and consulates worldwide. The embassy and consulate websites have helpful information about procedures, timelines, communities served, required documentation, and fees.

Letters from ACM

International registrants should be particularly aware and careful about visa requirements, and should plan travel well in advance. All visa inquiries must be handled by ACM Headquarters. Please send your request for a letter in support of a visa application to Ashley Cozzi (acozzi at acm.org), and include your name, mailing address, and fax number, as well as the name of the conference you are attending. (Authors of papers/posters should also include the title). Please note that ACM does not issue formal "letters of invitation" to any of its conferences.

Ada and Education

Public Ada Courses

From: Ed <colbert@abssw.com>
Date: Fri, 22 Feb 2008 17:54:54 −0800 (PST)
Subject: [Reminder] Public Ada Courses 3–7 March '08 in Carlsbad CA
Newsgroups: comp.lang.ada

Absolute Software will be holding a public Ada course during the week of 3 March in Carlsbad, CA. You can find a full description and registration form on our web-site, www.abssw.com. Click the Public Courses button in the left margin. (We also offer courses on software architecture-based development, safety-critical development, object- oriented methods, and other object-oriented languages.) [...]

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

SPARK Training

From: Praxis HIS — SPARKAda
Subject: SPARK Training
Date: March, 2008
URL: http://www.praxis-his.com/sparkada/training.asp

Public Course Dates for 2008 — UK

Course 1 — "Software Engineering with SPARK"

3rd – 6th March 2008 at the Praxis Offices in Bath

15th – 18th September 2008 at the Praxis Offices in Bath

Course 2 — "Black-Belt SPARK"

11th – 13th March 2008 at the Praxis Offices in Bath

23rd – 25th September 2008 at the Praxis Offices in Bath

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

Webminar: GPS InSight

From: AdaCore Press Center
Date: Thursday December 13, 2007
Subject: GPS InSight Webinar
RSS: http://www.adacore.com/2007/12/13/gps-insight-webinar/

GPS InSight Webinar

The archive of this webinar featuring a presentation and demo of GPS 4.2.0 is now available. Please click here to view it or visit:

http://www.adacore.com/home/gnatpro/webinars

[See also "Webminar: GNATbench" and "Webminar: Eclipse" in AUJ 28-3 (Sep 2007). —su]

Ada-related Resources

New Blog about Ada

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Mon, 21 Jan 2008 11:53:33 +0100
Subject: [blog] The Henn and Egg problem
Newsgroups: comp.lang.ada

There is a new entry on the Ada programming blog [...] which might be of interest.

http://ada-programming.blogspot.com/

PS: it is possible to add additional authors to the "Ada programming blog" — mail me if you are interested.

Ada Social networks

From: melampus <henssel@gmail.com>
Date: Tue, 22 Jan 2008 02:34:16 −0800 (PST)
Subject: Re: Social networks for Ada people
Newsgroups: comp.lang.ada

> If a social network dedicated to Ada programmers existed, how many of the people here would use it?

Something like Ohloh (http://www.ohloh.net/) would at least present Ada — and its fans — as "modern" and finding solutions to the current ITC demand and issues. Currently most ITC folks probally sees Ada as more Dino than T.Rex ;-)

We could also try to establish "open" Ada groups in Facebook, LinkedIn or other web waterholes.

From: Manuel Gomez <mgrojo@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: Social networks for Ada people
Date: Tue, 22 Jan 2008 11:58:48 −0800 (PST)

By the way, Ada is already in that site, see http://www.ohloh.net/languages/21

From: Manuel Gomez <mgrojo@gmail.com>
Newsgroups: comp.lang.ada

Subject: Re: Social networks for Ada people
Date: Tue, 22 Jan 2008 11:41:31 −0800 (PST)

> That's what the existing Ada wikis are. Here is a list:

English: http://en.wikibooks.org/wiki/Ada_Programming

English: http://ada.krischik.com/index.php

French: http://fr.wikibooks.org/wiki/Programmation_Ada/FAQ/

French: http://www.ada-france.org

I know these wikis, in fact I'm one of the contributors of Ada Programming wikibook. The problem with the wikibook is its perfectly defined scope, it's a wiki for writing a text-book. The site I was thinking about would probably have a wiki open for any subject related to Ada, but it should include other web 2.0 features so it can become the community-driven version of:

http://www.adapower.com/

http://www.adaworld.com/

For example, it should include uploading of code snippets, presentations (or embedded from something like

http://www.slideshare.net/tag/ada

submission of articles, etc. All of this content should be submitted under a clear open-content license, like GFDL, or Creative Commons.

FAQ fr.comp.lang.ada changes

From: Samuel Tardieu <sam@rfc1149.net>
Date: Sun, 2 Mar 2008 11:00:43 GMT
Subject: [FAQ] fr.comp.lang.ada
Newsgroups: fr.comp.lang.ada,fr.usenet.reponses
Summary: Questions fréquemment posées sur le groupe de discussion fr.comp.lang.ada, dédié au langage Ada

[Translated from French. —su]

WWW-Archive-Name: http://www.rfc1149.net/fcla/

Maintainer: sam@rfc1149.net (Samuel Tardieu)

Last-Modified: Wed Feb 21 11:18:00 CET 2007

Archive-Name: fr/comp/lang/faq-ada

FAQ fr.comp.lang.ada

After March 5, 2007, the fr.comp.lang.ada FAQ is collectively managed at:

http://fr.wikibooks.org/wiki/Programmation_Ada/FAQ/

Do not hesitate to consult and improve it.

Ada-related Tools

AdaCL 5.0.8 — Ada Class Library

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Sat, 08 Dec 2007 19:34:56 +0100
Subject: [Announcement] AdaCL 5.0.8 released
Newsgroups: comp.lang.ada

I have just released a new version of AdaCL which fixes the problems with newer GNAT compilers. At least when the garbage collector support is switched off it is now possible to compile AdaCL with GNAT.

To download and to see what AdaCL actually does see:

http://adacl.sourceforge.net/

Note that with Version 5 AdaCL is now strictly Ada 2005 and makes use of Ada 2005 features.

AdaCL will also be part of the next release of The GNU Ada Project.

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: Re: [Announcement] AdaCL 5.0.8 released
Newsgroups: comp.lang.ada
Date: Sat, 22 Dec 2007 14:05:56 +0100

I have just uploaded the Solaris 10 packages. Get it while it's hot:

http://gnuada.sourceforge.net/

[See also "AdaCL 4.2.0 — Ada Class Library" in AUJ 25-4 (Dec 2004) —su]

Strings Edit

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sat, 12 Jan 2008 20:14:06 +0100
Subject: ANN: Strings_Edit v2.0
Newsgroups: comp.lang.ada

The library provides a set of packages for formatting and text processing of:

- Integer numbers (generic, package Integer_Edit);
- Integer sub- and superscript numbers;
- Floating-point numbers (generic, package Float_Edit);
- Roman numbers (the type Roman);
- Strings;
- Ada-style quoted strings;
- UTF-8 encoded strings;
- Unicode case mappings;
- Wildcard pattern matching.

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

[...]

[See also same topic in AUJ 28-2 (Jun 2007) —su]

Simple components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 10 Feb 2008 21:37:04 +0100
Subject: ANN: Simple components v2.7
Newsgroups: comp.lang.ada

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

Changes to the version 2.5:

- Function Is_Empty was added to doubly-linked lists;
- Functions Erase and Take were added for doubly-linked webs and lists:
- Persistent storage packages interface was changed from Wide_String (UCS-2) to String (UTF-8) Unicode support;
- Persistent storage now supports hierarchical names of objects;
- Get_Class abstract operation was added to the persistent storage interface;
- The package Generic_Random_Sequence was added to provide random sequences of non-repeating numbers;
- The package Strings_Edit.Symetric_Serialization provides symmetric encryption and encoding of short plain strings, which can be used for storing user credentials.

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

AdaSubst & AdaDep

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 12 Feb 2008 17:31:59 +0100
Organization: Adalog
Subject: Updated versions of AdaSubst and AdaDep
Newsgroups: comp.lang.ada

Adalog is pleased to announce a new release of AdaSubst and AdaDep. No real change in functionality, but better packaging, doc now in info, html, and pdf formats, and makes use of the latest version of the components that are common with AdaControl (i.e. bugs fixed).

Download and more info from http://www.adalog.fr/compo2.htm

[See also same topic in AUJ 25-2 (Jun 2004) —su]

GNU Ada Compiler

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Sun, 09 Dec 2007 17:46:55 +0100
Subject: [Announcement] The GNU Ada Project Release 8
Newsgroups: comp.lang.ada

New with release 8 of The GNU Ada Project brings you:

1) GNAT/GCC 4.2.2
2) GNAT/GPL 2007-2
3) A GNAT Programming Studio for

GCC and GPL
4) AdaCL, the Ada Class Library

The GPS it the newest I could compile as there are compiler dependencies and the new GPS 4.2.0 will compile only with GNAT/PRO. This implies that I could not distribute the newest GtkAda and XMLAda as as an older GPS won't compiler with newer libraries.

As allways SuSE 10.3 x86_64 is the first to be released with other Systems following later. So stay tuned.

[See also same topic in AUJ 28-3 (Jun 2007) —su]

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Sun, 27 Jan 2008 13:28:42 +0100
Subject: [anouncement] MingGW release for the GNU Ada Project
Newsgroups: comp.lang.ada

[...]

This is the first release which includes the files needed to compile ASIS, GLADE and GPS later on [1].

Like the Unix Releases this release installs into /opt and therefore won't overwrite your default MinGW compiler.

The release is still not as complete as the Unix bases releases but it allready contains the Booch components and XMLAda.

Download from:

http://sourceforge.net/project/showfiles.php?group_id=12974&package_id=240621

http://sourceforge.net/project/showfiles.php?group_id=12974&package_id=260608

[1] Note to other maintainers: Those files are *not* included on “make install” — You have to add them yourself!

From: Simon Wright <simon.j.wright@mac.com>
Date: Sun, 27 Jan 2008 19:36:25 +0000
Subject: ANN: GNAT/GPL Solaris 10: 2007-solaris-x86
Newsgroups: comp.lang.ada

I've uploaded this to the GNU Ada project at SourceForge: http://sourceforge.net/project/showfiles.php?group_id=12974&package_id=260618 & release_id=571804

From: Simon Wright <simon.j.wright@mac.com>
Date: Mon, 14 Jan 2008 18:38:17 +0000
Subject: ANN: GNAT/GPL Mac OS X: 2007-tiger-ppc
Newsgroups: comp.lang.ada

I've uploaded this to the GNU Ada project at SourceForge:

http://sourceforge.net/project/showfiles.php?group_id=12974&package_id=258771

You might find its location at /opt/gnat-gpl-2007 unhandy, you may find gnatfe helpful:

http://sourceforge.net/project/showfiles.php?group_id=12974&package_id=258764

AVR-Ada

From: Rolf Ebert <rolf.ebert@gmx.net>
Date: Thu, 20 Dec 2007 23:34:41 −0800 (PST)
Newsgroups: comp.lang.ada
Subject: [Ann] AVR-Ada V0.5.2 released

We are proud to announce a new release of AVR-Ada, one of the first GCC based Ada compilers targeting 8-bit microcontrollers.

You get the project description and some documentation at

http://avr-ada.sourceforge.net/

The Sourceforge development pages with the download section are at

http://www.sourceforge.net/projects/avr-ada/

AVR-Ada is available in source and binary form for Windows. The binary packages of the cross compiler hosted on Windows is now part of the just released WinAVR tool suite.

For any questions please join the mailing list at

http://lists.sourceforge.net/mailman/listinfo/avr-ada-devel

It has quite low traffic.

Please use SF's bug reporting and feature request system for guiding future development of AVR-Ada.

[See also same topic in AUJ 27-2 (Jun 2006) —su]

Gela — BSD compiler in development

From: Vadim Godunko <vgodunko@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: Restricted or no run time in Ada
Date: Thu, 3 Jan 2008 12:05:50 −0800 (PST)

> GNAT (non-PRO) is the only one that is free and open source.

Yet another open source (under BSD license!) Ada compiler:

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

[The following description is from the web page —su]

Gela — Ada Support for TenDRA

The goal of Gela project is creation of portable Ada compiler.

TenDRA is a C compiler, with C++ STL support forthcoming. The original Crown copyright from DERA is still present, and further expansion of TenDRA is under the BSD License.

TenDRA uses the TenDRA Distribution Format (TDF) as its intermediate language. It is based on (X)ANDF (Architecture Neutral Distribution Format) which evolved from TDF. ANDF focuses on abstracting high level languages instead of assembler languages, as is common with most compilers. This makes TenDRA a powerful tool in code verification and checking.

TenDRA Goals:

- To continuously produce correct code.
- To continuously improve the performance of the compiler and resulting code, unless it would jeopardise the point above.
- To create tools that facilitate programming, not to have programming facilitate the tools.
- To be a friendly competitor to GCC in order to get a best-of-breed compiler.

From: Maxim Reznik <reznikmm@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: Restricted or no run time in Ada
Date: Fri, 4 Jan 2008 06:13:19 −0800 (PST)

[...] Today most usable part of Gela is Gela-ASIS. This is target independent ASIS implemented from scratch. It implement most of ASIS for Ada 95 queries and 12 extension to support Ada 2005 according to SI99 proposals.

From: Maxim Reznik <reznikmm@gmail.com>
Date: Fri, 4 Jan 2008 01:23:18 −0800 (PST)
Subject: Re: Restricted or no run time in Ada
Newsgroups: comp.lang.ada

[...] Of course Gela project is far away of completion and not yet ready to be used in your project, but code it can generate for now not much worse than one of others compilers.

GTKAda contributions

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Subject: ANN: GtkAda contributions 2.1
Newsgroups: comp.lang.ada
Date: Sun, 10 Feb 2008 21:42:26 +0100

The packages extend GtkAda, an Ada bindings to GTK+. It deals with the following issues:

- Tasking support;
- Custom models for tree view widget;
- Custom cell renders for tree view widget;
- Multi-columned derived model;
- Extension derived model (to add columns to an existing model);
- Abstract caching model for directory-like data;
- Tree view and list view widgets for navigational browsing of abstract caching models;
- File system navigation widgets with wildcard filtering;

- Resource styles;
- Capturing resources of a widget;
- Embeddable images;
- Some missing subprograms and bug fixes;
- Measurement unit selection widget and dialogs;
- Improved hue-luminance-saturation color model;
- Simplified image buttons and buttons customizable by style properties;
- Controlled Ada types for GTK+ strong and weak references;
- Simplified means to create lists of strings.

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

[...]

[See also same topic in AUJ 28-4 (Dec 2007) —su]

QtAda binding

From: Vadim Godunko <vgodunko@gmail.com>
Date: Tue, 12 Feb 2008 13:20:38 −0800 (PST)
Subject: Announce: QtAda 1.0.2
Newsgroups: comp.lang.ada

We are pleased to announce QtAda 1.0.2 release. This release includes fixes for critical bugs and workarounds for the GNAT compiler's bugs. Full list see at the end of this mail.

QtAda is an Ada2005 language bindings to the Qt libraries and a set of useful tools. QtAda allows easily to create cross-platform powerful graphical user interface completely on Ada 2005. QtAda applications will work on most popular platforms — Microsoft Windows, Mac OS X, Linux/Unix — without any changes and platform specific code. QtAda allows to use all power of visual GUI development with Qt Designer on all software lifecycle stages — from prototyping and up to maintenance. QtAda is not just a bindings to the existent Qt widgets, it also allows to develop your own widgets and integrates it into the Qt Designer for high speed visual GUI development.

QtAda can be downloaded from:

http://www.qtada.com/

[...]

[See also same topic in AUJ 28-4 (Dec 2007) —su]

AutoIT — Automated GUI Testing

From: Per Sandberg <per.sandberg@bredband.net>
Date: Thu, 07 Feb 2008 06:38:14 +0100
Subject: ada-AutoIT 1.5.3 Released
Newsgroups: comp.lang.ada

ada-AutoIT 1.5.3 is released

Small cleanups and bumped to AutoIT 3.2.10.0

Win32 only !

AutoIT Homepage:

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

SourceForge

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

[See also same topic in AUJ 27-4 (Dec 2006) —su]

TclAdaShell

From: Simon Wright <simon.j.wright@mac.com>
Date: Sun, 03 Feb 2008 22:02:39 +0000
Subject: ANN: TclAdaShell release
Newsgroups: comp.lang.ada

I've released two packages, which you can find here:

http://sourceforge.net/project/showfiles.php?group_id=164395

Under 'source' is a release for people who have a full development environment. As is, it works with GNAT GPL 2007 (will work with earlier versions, with some makefile tweaks).

Under 'windows' is a Windows binary release (made on Windows 2000): GNAT GPL 2007, ActiveState's Tcl.

I haven't made binary releases for other platforms, mainly because there are so many of them.

More info at http://tcladashell.wiki.sourceforge.net/

Hibachi plans

From: Tom Grosman <grosman@aonix.fr>
Date: February 23, 2008 00:53
Subject: RE: [hibachi-dev] Hibachi Plans Questions
newsgroup: eclipse.tools.hibachi

[...] There is no effort to include a compiler with Hibachi. This is not possible for several reasons. First, everything released under the Hibachi project must meet Eclipse's rigorous Intellectual Property rules. So for instance, even the GNAT compiler, which is under GPL, couldn't be distributed as part of Hibachi since GPL is not compatible with EPL. There are other reasons why a compiler is not included as part of Hibachi, but the one I listed is a deal breaker, so no need to go on.

That said, there will probably be a release of an Eclipse based Ada development environment bundle including Hibachi, the GNAT toolchain and anything else needed to do Ada development with a single download. However, it will not be part of the Hibachi project. It would be available via a non-Eclipse download site. This is actually what Doug Schaefer (the CDT project lead) has done for C/C++ with Wascana (see http://wascana.sourceforge.net/).

[See also "Hibachi official Eclipse Open Source Project" and "Aonix — Eclipse Hibachi Project Unites Ada Suppliers in Common Environment" in AUJ 28-4 (Dec 2007). —su]

Hibachi sources available

From: Tom's Hibachi musings
Date: January 17, 2008
Subject: Sources available!
RSS: http://hibachitom.blogspot.com/2008/01/sources-available.html

The Hibachi sources are now available on the Eclipse server. There have already been a few folks who have downloaded them and built Hibachi. Getting the sources through the parallel IP process was not a piece of cake. We had to replace the parser generator, and leave out the pretty printer. Even though the authors of the pretty printer gave their written consent for us to use it, because it had been released elsewhere under GPL (which is not Eclipse compliant), we couldn't use it. The IP process also brought up such issues as comments that suggested use of a possibly non-compliant library (not the actual use of the library, mind you).

I suppose it's a good thing that the legal team at Eclipse is so thorough. In the end it will help insure widespread acceptance, but in the short run, it adds a non-trivial delay to the moment when a piece of code becomes available for use.

UnZip-Ada

Date: Fri, 11 Jan 2008 06:46:38 +0100
From: Gautier de Montmollin <gdemont@hotmail.com>
Newsgroups: comp.lang.ada
Subject: Ann: UnZip-Ada v.23

Here is some progress on the UnZip & Zip libraries...

You find UnZip-Ada there:

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

and there:

http://homepage.sunrise.ch/mysunrise/gdm/unzipada.htm

[...]

[See also same topic in AUJ 28-2 (Jun 2007) —su]

Ada-related Products

AdaCore — Plans for 2008

From: Jamie Ayre <ayre@adacore.com>
Subject: [AdaCore] Happy Ada programming in 2008
Date: Mon, January 7, 2008 9:07 am
To: announce@adacore.com

Happy new year, and best wishes for successful Ada programming in 2008! I wanted to take this opportunity to give you an idea about what we are planning for the coming year. This is tentative of course, since we constantly redefine our development plans in light of customer wishes and needs.

On the compiler front, the implementation of Ada 2005 features is essentially complete, and what is very encouraging is that we have several users who have been banging away at these new features furiously, in some cases with frighteningly complex programs. This means that the implementation has matured very rapidly, and the new 6.1.1 release is already a very solid implementation of Ada 2005. So you can venture in this direction with confidence.

We are busy planning a number of interesting new features for the compiler, including an implementation of precondition and postcondition pragmas. Keep a watch on the websites at the development center for latest news, and remember, if you see a feature that sounds interesting to try out, you can always request a wavefront that incorporates all the latest fixes and features.

Many new ports of the GNAT Pro technology have appeared recently and more will appear in the near future, including ports for Microsoft .NET, Nucleus OS, Ardence RTX, VxWorks 653 version 2.2, VxWorks 6.5, LynxOS 5, and Windows Vista. We intend to continue to provide versions of GNAT Pro for all popular architectures and operating systems.

We are also pursuing incorporation of the latest GCC and GDB technology, including GCC 4.3 and GDB 6.6, to enable our customers to benefit from the latest developments in the GCC/FSF community.

In 2007 we refocused our safety critical products to better serve our users with the new High-Integrity Family. The first member of this family is the "GNAT Pro High-Integrity Edition for DO-178B" for embedded safety-critical application development. In 2008 we will be enhancing the second member of this family, the "GNAT Pro High- Integrity Edition for Servers". This Edition is designed to address high-integrity requirements in server contexts as opposed to embedded environments, and will include the high-integrity cross run-time libraries available on host configurations to allow easier host-based testing of embedded applications.

In the tools area, we are moving energetically into the area of qualified tools, including GNATstack (static stack analysis) and GNATcheck (coding/style conformance checking). We have also incorporated GPROF (profiling) and GCOV (coverage analysis) into GNAT Pro as fully supported elements of the technology.

In the area of IDE's, AdaCore supports both the GPS and GNATbench environments. We are positioned to support both of these environments on a permanent long term basis. This dual support is made practical by the structure in which the great majority of the Ada functionality of both these IDE's is embodied in a common library shared by both implementations. 2008 will see multiple releases of both IDE's with many new features.

Recognizing the continued importance of mixed language programming, we will be releasing comprehensive Java interfacing technology allowing mixing of Ada and Java, including the use of binding generators in both directions. In addition, our new GPRBUILD tool brings the power of project file-driven builds to multi-language programs.

Another important new direction is the first release of the "AdaCore Reusable Components". These components will be extracted from our internal technology (such as GPS and GNAT Tracker) to let our customers benefit from high-level libraries that solve common programming challenges.

Again, we wish you a productive new year, and we look forward to working with you to ensure the success of your Ada projects.

Robert Dewar

AdaCore President / CEO

AdaCore — GPS 4.2.0

From: AdaCore Press Center
Date: December 10, 2007
Subject: GPS 4.2.0
RSS: http://www.adacore.com/2007/12/10/gps-420/

AdaCore is pleased to announce the immediate availability of GPS 4.2.0. New functions in GPS 4.2 include:

- Graphical support for code coverage (gcov)
- Improved documentation generation with faster, improved HTML output using CSS and Javascript
- Enhanced code completion, including support for the Object.Method syntax as provided in Ada 2005.
- Full ability to manage files and directories from GPS
- Source editor improvements better tooltips, source navigation and indentation
- Improved handling of dispatching calls and primitives, enabling better understanding (prior to run time) of which subprograms will be executed

New plug-ins, including:

- Support for code verification through gnatcheck
- Support for addr2line
- Listing of unused entities (replaces gnatxref)
- Display of dependency paths across files
- Ability to cut/copy/paste in contextual menu
- Recomputation of Ada cross references

GPS 4.2.0 is compatible with GNAT Pro versions 3.16a1 up to 6.1.

[See also "AdaCore — GPS 4.1.3" in AUJ 28-4 (Dec 2007) —su]

AdaCore — GNATcheck

From: AdaCore Press Center
Date: Wednesday January 30, 2008
Subject: Coding Standard Verification Tool Eases DO-178B Compliance
RSS: http://www.adacore.com/2008/01/30/coding-standard-verification-tool-eases-do-178b-compliance/

TOULOUSE, France and NEW YORK — January 30, 2008 — Embedded Real-time Software (ERTS) Conference — AdaCore, provider of the highest quality Ada tools and support services, today announced the availability of GNATcheck, an integrated coding standard verification tool within the GNAT Pro development environment. GNATcheck meets the growing need for automated verification in safety-critical avionics systems, particularly those systems that need to satisfy the DO-178B standard. Developed by RTCA and EUROCAE, DO-178B defines the guidelines for development of aviation software in both the US and Europe and is being increasingly adopted by other related sectors, such as air traffic control and military applications.

AdaCore's GNATcheck is an extensible rule-based tool with an easy-to-use interface. It allows developers to completely define a coding standard (referred to as a "Software Code Standard" in DO-178B) as a set of rules, for example a subset of permitted language features. It verifies a program's conformance with the resulting rules and thereby facilitates demonstration of a system's compliance with DO-178B.

"The combination of the Ada language (an international standard), GNATcheck, and additional constraints and reporting inside GNAT Pro, provides a comprehensive solution for avionics developers," said Robert Dewar, President and CEO of AdaCore. "Ada has already been used in many safety-critical systems, such as the Boeing 787 and C-130 AMP, Airbus A380 and Eurofighter among others."

"With software innovation powering today's successful aircraft, automatic coding standard verification is becoming more and more important," commented

Cyrille Comar, Managing Director, AdaCore Europe. "The highly structured nature of the Ada language makes it a natural choice for avionics development, and by adding our own enhancements within GNAT Pro we can offer the most complete and integrated solution for coding standard verification compliant with DO-178B requirements."

The key features of GNATcheck include:

- An integrated Ada Restrictions mechanism for banning specific features from an application. This can be used to restrict features, such as tasking, exceptions, dynamic allocation, fixed or floating point, input/output and unchecked conversions

- GNAT Pro specific Restrictions, which complement Ada's set of restrictions, such as those banning the generation of implicit loops or conditionals in the object code, or the generation of elaboration code

- Additional rules based on Ada semantics specification developed following extensive customer input, including detailed issues, such as ordering of parameters, normalized naming of entities and subprograms with multiple returns

- Easy-to-use interface for creating and using a complete coding standard

- Generation of project-wide reports, including evidence of the level of compliance to a given coding standard

- Over 30 compile time warnings from GNAT Pro that detect typical error situations, such as local variables being used before being initialized, incorrect assumptions about array lower bounds, infinite recursion, incorrect data alignment, and accidental hiding of names

- Style checks that allow developers to control indentation, casing, comment style, and nesting level

Work on qualifying GNATcheck as a verification tool (in a DO-178B context) is in progress. After this work is completed, GNATcheck's status as a qualified tool will allow the evidence that it generates to be used as part of a system's certification.

Pricing and Availability

GNATcheck is currently available as part of the GNAT Pro subscription. Please contact AdaCore (sales@adacore.com) for the latest information on pricing and supported configurations.

About AdaCore

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

Ada and GNAT Pro continue to see growing usage in high-integrity and safety-certified applications, including commercial aircraft avionics, military systems, air traffic management/control, railroad systems, and medical devices, and in security-sensitive domains such as financial services. AdaCore has North American headquarters in New York and European headquarters in Paris. www.adacore.com

AdaCore — GNATbench 2.1.0

From: Jamie Ayre <ayre@adacore.com>
Subject: [AdaCore] Announcing the availability of GNATbench 2.1.0 for Eclipse
Date: Fri, February 29, 2008 6:56 am
To: announce@adacore.com

AdaCore is pleased to announce GNATbench 2.1.0, a major enhancement to the GNAT Pro Eclipse-based plug-in, for the following platforms:

sparc-solaris

x86-linux

x86_64-linux

x86-windows

p55-elf-windows

ppc-elf-solaris

ppc-elf-windows

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

GNATbench 2.1.0 introduces the following features among others:

Project Management and Presentation

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

Language-Sensitive Editor Enhancements

- "Quick Fix" for Ada
- Smart Space Key
- Automatic Construct Closing
- Smart Tab Key
- Smart Enter Key
- Text and Comment Lines Refilled Against Margin
- Special Coloring for "Annotation Comments"
- The "Show In" Contextual Menu Entry Supported
- Improved Comment Block Selection
- Standard Parenthesis Highlighting
- Integration of Code Formatting Features
- Recent Chosen Completions Displayed at the Top

Additional Wizards

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

Builder Enhancements

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

Source Code Navigation Enhancements

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

Miscellaneous Improvements

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

GNATbench 2.1.0 can be downloaded as usual using GNAT Tracker. As always, for questions, or to inform us of issues that you encounter, please let us know through the GNAT Tracker report facility or by email to the usual report@adacore.com address.

[See also "AdaCore — GNATbench Eclipse Plug-in" in AUJ 28-2 (Jun 2007) —su]

AdaCore — GNAT Pro 6.1.1

From: AdaCore Press Center
Date: Wednesday March 5, 2008
Subject: Latest GNAT Pro Release Adds New Platforms, New Tools
RSS: http://www.adacore.com/2008/03/05/gnatpro-611-release/

Powerful Ada Development Environment Enhanced for High-Integrity Systems

NEW YORK and AMSTERDAM, Netherlands, March 5, 2008 — Avionics 2008 — AdaCore, provider of the highest quality Ada tools and support, today announced the company's twelfth annual release of its signature GNAT Pro Ada development environment. GNAT Pro 6.1.1 offers more than 150 new features and is available on the largest set of supported platforms in the industry — 44 configurations (including 28 cross compilers) on 79 different operating system versions, including multiple versions of Windows, Linux and Solaris.

"Our annual release cycle is a major part of our commitment to make Ada the development language of choice for long-lived, critical systems," said Cyrille Comar, Managing Director, AdaCore

Europe. "Our expert team of engineers is constantly working on enhancements and new tools for GNAT Pro that will help customers for years to come. We are particularly focused on high-integrity systems, where many of Ada's advantages stand out."

"Ever since our company was founded, our customers have come to expect the level of quality and front-line support that our solutions provide," said Robert Dewar, President and CEO of AdaCore. "We pride ourselves on being able to offer regular enhancements, many of which originate as customer requests. After a rigorous quality assurance process, we have integrated many of these features as part of this latest version of the GNAT Pro tool suite."

GNAT Pro 6.1.1's new features include:

- High-Integrity versions for VxWorks 6, including the Ravenscar profile
- Thread-safe profiling with gprof, a tool currently available for GNAT Pro on several platforms
- Increased Ada support in gcov, a coverage analysis tool
- Enhanced tools such as gnatcheck, gnatpp, and gnatmetric, to support a wider variety of coding styles and coding standards
- New warnings to help programmers detect errors earlier
- An upgraded debugging engine
- Improved robustness and efficiency for Ada 2005 features
- Better real-time support on win32 platforms
- Fully-integrated Windows .NET framework support

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 offers the flexibility and freedom associated with open source development, together with the assurance that comes from knowing that all tools go through a rigorous quality assurance process. GNAT Pro is based on the widely used GCC technology and is backed by rapid and expert support service.

Pricing

Pricing for GNAT Pro subscriptions starts at $14,000. Please contact AdaCore (sales@adacore.com) for the latest information on pricing and supported configurations.

[See also "AdaCore — GNAT Pro 6.0.1" in AUJ 28-1 (Mar 2007) —su]

Adalog — AdaControl

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 05 Feb 2008 18:44:40 +0100
Subject: AdaControl 1.8r7 released
Organization: Adalog
Newsgroups: comp.lang.ada

Adalog is pleased to announce the release of a new version of AdaControl. This version features improvements to GPS integration and, of course, plenty of new rules. There are now 317 possible checks, at the latest count!

The user guide has also been improved, as well as examples of rules files (including equivalences to Gnatcheck).

And of course, everything is still GMGPL, with commercial support available from Adalog.

Download from http://www.adalog.fr/adacontrol2.htm

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

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Fri, 08 Feb 2008 13:47:53 +0100
Subject: Small glitch with AdaControl fixed
Organization: Adalog
Newsgroups: comp.lang.ada

The new GPS interface requires GPS 4.2, which made it non-functional with the version of GPS distributed with Gnat/GPL2007.

I made a new release available (tagged 1.8r8) that provides the old interface for those who have not upgraded their GPS.

There is no change in AdaControl itself.

Aonix — ObjectAda 8.3 for Solaris

From: Aonix Press releases
Subject: Aonix's ObjectAda Brings Eclipse to Sun Solaris Platforms
Date: January 28, 2008
URL: http://www.aonix.com/pr_01.28.08.html

Eclipse enables best-selling Ada technology to better serve large project groups

San Diego, CA, January 28, 2008

Aonix®, a provider of solutions for safety- and mission-critical applications, announced the release of ObjectAda for Solaris products. ObjectAda V8.3 for Sun's popular Solaris platforms running on SPARC and Intel processors provides a complete enterprise-level environment for the development of native Unix applications using the Ada programming language. These latest releases integrate AonixADT (Ada Development Toolkit), an Eclipse-based development environment, into ObjectAda, providing developers with access to the broad range of tools available through the Eclipse framework.

Aonix offered the first commercial Eclipse plug-in for ObjectAda in 2004. To further this initiative, Aonix has contributed the AonixADT source code as the baseline code for Hibachi, a new open-source Ada development tooling project that parallels and complements CDT, the C/C++ development tooling project. Both of these projects will provide a multilanguage native embedded software project for developers.

In addition to providing access to the broad spectrum of Eclipse-based tools, ObjectAda for Solaris supports a new dynamic debugging facility. Developers now have run-time debug capabilities and can attach the debug facility in the toolkit to an already running process in the application. Dynamic debug extends the already powerful debug capability provided in all Aonix native and cross-development products.

"As a long-standing leader in development tools for Unix environments, we are pleased to provide technology that advances methodologies and improves usability of development tools for our customers," noted Gary Cato, director of strategic alliances at Aonix. "Mission-critical systems are hard enough to develop without spending time and energy dealing with non-standard tools. ObjectAda with Eclipse toolset support provides significant additional development tools while minimizing toolset integration and learning-curve overhead"

ObjectAda® for Solaris comes with both a graphical and command-line interface, integrated language-sensitive editor, lightweight source-based library model, and industry-leading compilation speed. The ObjectAda for Solaris compilation system is composed of the editor, source-code browser, compiler, debugger, and full library manager.

Optional package upgrades can be added to the basic compiler development package such as the ObjectAda Project Pack that contains the AdaNav™ toolset. AdaNav provides complete system HTML source-navigation capabilities as well as call- and unit-tree graphical reporting and automatic data dictionary generation. The AdaNav profiler also offers run-time performance reporting to identify application hot spots.

A second package, ObjectAda Test Pack can also be added atop ObjectAda Project

Pack on the SPARC/Solaris platform that provides VectorCast/Ada a world-class testing tool that significantly reduces the time, effort, and cost associated with testing Ada software components necessary for validating safety- and mission-critical systems.

Shipping and Availability

ObjectAda for SPARC/Solaris and ObjectAda for Intel/Solaris are immediately available. Prices start at $8,000 for a single seat license. Quantity discounts are available.

About Aonix

Aonix offers mission- and safety-critical solutions primarily to the military and aerospace, telecommunications and transportation industries. Aonix delivers the leading high-reliability, real-time embedded virtual machine solution for running Java™ programs deployed today and has the largest number of certified Ada applications at the highest level of criticality. Headquartered in San Diego, CA and Paris, France, Aonix operates sales offices throughout North America and Europe in addition to offering a network of international distributors. For more information, visit www.aonix.com.

[See also “Aonix — ObjectAda RAVEN for VxWorks 653” in AUJ 28-4 (Dec 2007) —su]

DDC-I — OpenArbor for LynxOS-178

From: DDC-I Press releases
Subject: DDC-I Announces Mixed Language Development Support for LynuxWorks' FAA Certified LynxOS-178 RTOS
Date:January 7, 2007
URL: http://www.ddci.com/display_news_item-filename-news_Mixed_Language_Development_Support_release.htm

OpenArbor™ developers can now deploy safety-critical applications combining C, C++, and Ada on DO-178B-compliant systems running FAA-certified LynxOS-178

Phoenix, AZ. January 7, 2007. DDC-I, a leading supplier of development tools and engineering services for safety-critical applications, today announced that its Eclipse-based OpenArbor mixed-language development environment now supports LynuxWorks' FAA Certified LynxOS-178 real-time operating system. OpenArbor developers working with any combination of Ada, C, and Embedded C++ (EC++) can now deploy their safety-critical applications on LynxOS-178 target systems certifiable to DO-178B Level A, the FAA's highest level of safety criticality.

“Safety-critical software development and deployment are our bread and butter,” said Bob Morris, president and CEO of DDC-I. “LynxOS-178 provides an excellent platform for deploying safety-critical C, Embedded C++, and Ada, developed using OpenArbor, particularly for systems requiring FAA certification.”

“Open Arbor's open Eclipse framework and mixed language capability make it a great complement to our Luminosity development suite, and now offers the strongest set of integrated tools for developing safety-critical applications on target systems running LynxOS-178,” said Joe Wlad, director of marketing at LynuxWorks. “OpenArbor developers can now deploy their mixed-language applications on an ironclad, memory-partitioned RTOS with a fast track to FAA certification.”

LynxOS-178 is a real-time operating system designed to fulfill the stringent needs of multithreaded and multiprocess applications in safety-critical real-time systems. LynxOS-178 enhances safety and security by using Virtual Machine (VM) brick-wall partitions that prevent system events in one RTOS partition from interfering with events in another. Effectively, each partition behaves as if it were running on its own separate computer.

LynxOS-178 is the first and only commercial hard real-time operating system certified to DO-178B level A that combines the interoperability benefits of POSIX with support for the ARINC 653 APplication EXecutive (APEX). Available for both Pentium and PowerPC platforms, LynxOS-178 is also the first and only time- and space-partitioned, FAA-accepted Reusable Software Component (RSC). To speed the FAA certification process, LynxOS-178 provides complete DO-178B documentation, including an artifacts package. The Luminosity development environment for LynxOS-178 is also Eclipse-based, and by using a pristine Eclipse framework it can easily integrate with the OpenArbor environment from DDC-I via the standard Eclipse plug-in mechanism.

OpenArbor is a mixed-language, object-oriented IDE for developing and deploying real-time, safety-critical applications. The core environment combines optimizing compilers and libraries for C and Embedded C++ with the SCORE® mixed-language debugger.

The SCORE debugger features an intuitive multi-window GUI, project management support, and automated build/make utilities. SCORE's symbolic debugger recognizes C/EC++, Ada and Fortran syntax and expressions, and can view objects, expressions, call chains, execution traces, interspersed machine code, machine registers, and program stacks.

OpenArbor provides separate Eclipse plug-ins for Ada development. The Ada compiler plug-in, known as SCORE-Ada, features an optimizing Ada compiler and supports full Ada-level debugging, including constraints, attributes, tasking, exceptions, break-on-exception and break-on-tasking events. The Ada debugger plug-in supports true mixed language debugging in a single session, making it easy to debug applications written in multiple languages. The debugger is non intrusive, can debug at the source or machine level, and can be enabled without changing the generated code.

OpenArbor is available immediately for LynxOS-178. Pricing for the core configuration starts at $5,000.

About DDC-I, Inc.

DDC-I, Inc. is a global supplier of software development tools, custom software development services, and legacy software system modernization solutions, with a primary focus on safety-critical applications. DDC-I's customer base is an impressive “who's who” in the commercial, military, aerospace, and safety-critical industries. DDC-I offers compilers, integrated development environments and run-time systems for real-time Java, C, Embedded C++, Ada, and JOVIAL application development. For more information regarding DDC-I products, contact DDC-I at 1825 E. Northern Ave., Suite -125, Phoenix, Arizona 85020; phone (602) 275-7172; fax (602) 252-6054; e-mail sales@ddci.com

[See also “DDC-I — OpenArbor Eclipse Development Suite” in AUJ 28-4 (Dec 2007) —su]

Rapita Systems — RapiTime 1.3

From: Rapita Systems News
Subject: RapiTime Version 1.3
Date: 8 October, 2007
RSS: http://www.rapitasystems.com/node/150

Rapita is pleased to announce the latest release of RapiTime, version 1.3. This release provides many new features including:

- Enhanced support for function pointers. RapiTime can now perform an automatic discovery of the targets of function pointers.

- Generic text trace filters. Allows the processing of a wide range of text based trace formats (Debugger, Logic Analyzers, etc.).

- All new RapiTime report browser based on Eclipse technology.

From: Rapita Systems News
Subject: RapiTime Code Analysis Tool Gets Eclipsed

Date: 17 October, 2007
RSS: http://www.rapitasystems.com/node/151

RapiTime 1.3 version to be demonstrated at UK Embedded System Show

Rapita Systems have announced that their acclaimed software analysis tool RapiTime has undergone a number of significant updates, a key feature being that RapiTime is now integrated with the industry standard Eclipse IDE. Rapita's state-of-the-art RapiTime worst-case execution time analysis solution has enjoyed a great deal of success and recognition since its inception and the additional features are expected to broaden its appeal amongst embedded software developers.

The new RapiTime 1.3 version includes full Eclipse based RapiTime report visualisation, fast display of very large reports, integration with standard C Eclipse based source code editor and source code worst-case path colourisation, along with a number of other new features. Rapita's ambition for RapiTime version 1.3 was to enhance usability and also to make RapiTime more easily integrated into existing development environments.

"The RapiTime Eclipse report plug-in makes integration of RapiTime with other tools very easy," said Dr. Guillem Bernat, CEO of Rapita Systems Ltd. "It also provides greater flexibility in presenting and manipulating timing data. We are confident the new RapiTime will be appreciated by engineers working to develop responsive and reliable code."

Other features that contribute to the new RapiTime are customisable report tables, improved support for RTOS integration, support for logic analyser based tracing and improved display and handling of execution time profiles. In the UK, RapiTime is sold through European embedded systems tools specialist, SDC Systems, who will be exhibiting the Rapita toolset at the Embedded Systems Show.

"The RapiTime product has been exciting for us," said Stuart Parker. "It's proven to be best in its class and the new features will certainly put it further ahead in its field. RapiTime is saving some of our customers a lot of development time and energy, and helping to create far more robust embedded software. It's proving to be a product that can really make a difference."

SDC can be found on stand 520 and will also be exhibiting a range of embedded development products that include embedded Linux from MontaVista, embedded BIOS solutions from General Software and embedded GUI tools from Tilcon.

About SDC Systems

SDC Systems Limited is a leading European distributor of embedded development software, tools and hardware. Focused on innovative and leading edge technology, SDC strives to provide "technology that makes a difference", technology that will positively impact the development process and the quality and functionality of the final product. With many years of experience and talented engineering support, SDC Systems work closely with their clients to provide an important source of embedded expertise and products. www.sdcsystems.com

About Rapita Systems Ltd

Rapita Systems Ltd. is a specialist in the worst-case execution time (WCET) analysis and simulation of real-time embedded systems for the avionics, automotive and telecommunications markets. Its innovative RapiTime product makes Rapita Systems the leader in measurement based on-target WCET analysis solutions. www.rapitasystems.com

Vector Software — VectorCAST for NEC V850

From: Vector Software Press Releases
Subject: Vector Software Provides Automated Software Test Solution for NEC's V850 Development Environment
Date: December 3, 2007
URL: http://www.vectors.com/pdf/vc_pr_nec_v850.pdf

North Kingstown, RI — December 3, 2007 — Vector Software, Inc., a world leader in the embedded software test tool market, today announced the integration of their VectorCAST test tool with the NEC V850 compiler environment. The joint offering will provide developers using the NEC V850 environment with an automated unit and integration test capability allowing faster time to market with lower cost and better reliability.

Vector Software's VectorCAST™ is a world-class integrated software test solution that automates the tasks associated with testing software components for C/C++, Embedded C++, and Ada83/Ada95 programs. Automation includes: complete test harness construction (stubs and drivers), test generation, test execution, code coverage analysis, regression testing and static measures for code complexity, basis path analysis, and coding standards enforcement. VectorCAST enables companies to significantly reduce the time, effort and cost to validate safety, mission, and business-critical systems.

"At Vector Software, we are committed to delivering the industry-leading verification tools that developers of embedded software applications need," said Bill McCaffrey, director of marketing at Vector Software. "This integration of VectorCAST™ with the NEC V850 compiler environment provides several of our largest customers with the ability to automate unit, integration testing, and code coverage on the NEC v850 chip."

About NEC Electronics

NEC Electronics Corporation (TSE: 6723) specializes in semiconductor products encompassing advanced technology solutions for the high-end computing and broadband networking markets, system solutions for the mobile handset, PC peripherals, automotive and digital consumer markets, and platform solutions for a wide range of customer applications. NEC Electronics Corporation has 25 subsidiaries worldwide including NEC Electronics America, Inc. (www.am.necel.com) and NEC Electronics (Europe) GmbH (www.eu.necel.com). For additional information about NEC Electronics worldwide, visit www. necel.com.

About Vector Software

Vector Software, Inc, is a leading independent provider of automated software testing tools. Vector Software's VectorCAST line of products reduces the burden placed on individual developers by automating and standardizing application-component testing. The VectorCAST tools support the C, C++, Ada83, and Ada95 programming languages.

The market focus of Vector Software is on companies developing embedded systems for aerospace, military, medical, telecom, and process-control applications.

Vector Software's Product Family

VectorCAST/C++

VectorCAST/Ada

VectorCAST/RSP

VectorCAST/Cover

Modified Condition / Decision Coverage (MC/DC) module

DO-178B Qualification Packages

[See also "Vector Software — VectorCAST 4.0" in AUJ 27-2 (Jun 2006), and "Vector Software — VectorCAST for ARM" in this issue — su]

Vector Software — VectorCAST for ARM

From: Vector Software Press Releases
Date: January 31, 2008
Subject: Vector Software Adds Automated Test Capability to the ARM Compiler Environment
URL: http://www.vectors.com/pdf/vc_pr_arm1.pdf

North Kingstown, RI — January 31, 2008 — Vector Software, Inc., a world leader in the embedded software test tool market, today announced the integration of their VectorCAST™ test tool with the ARM compiler environment. The joint offering will provide developers using the ARM environment with an automated unit, integration, and system test capability allowing faster time to market with lower cost and better reliability.

Vector Software's VectorCAST™ is a world-class integrated software test solution that automates the tasks associated with testing software components for C/C++, Embedded C++, and Ada 83/Ada 95 programs. Automation includes: complete test harness construction (stubs and drivers), test generation, test execution, code coverage analysis, regression testing and static measures for code complexity, basis path analysis, and coding standards enforcement. VectorCAST™ enables companies to significantly reduce the time, effort and cost to validate safety, mission, and business-critical systems.

"The integration of our VectorCAST™ product with the ARM tool chain was the result of a request from a major semiconductor company," said Bill McCaffrey, Director of Marketing at Vector Software. "This integration of VectorCAST™ with the ARM compiler environment for ARM7 and ARM9 further illustrates the importance that Vector places on delivering our state of the art testing solutions to the most popular development environments."

About ARM

ARM designs the technology that lies at the heart of advanced digital products, from wireless, networking and consumer entertainment solutions to imaging, automotive, security and storage devices. ARM's comprehensive product offering includes 16/32-bit RISC microprocessors, data engines, 3D processors, digital libraries, embedded memories, peripherals, software and development tools, as well as analog functions and high-speed connectivity products.

[See also "Vector Software — VectorCAST 4.0" in AUJ 27-2 (Jun 2006), and "Vector Software — VectorCAST for NEC V850" in this issue —su]

Ada and GNU/Linux

Debian — Transition to GCC 4.3

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Newsgroups: comp.lang.ada
Subject: Ada in Debian: transition to GCC 4.3 for Lenny
Date: Tue, 5 Feb 2008 02:03:49 −0800 (PST)

My last status report was back in June 2007 so I think I should keep everyone posted.

gnat-4.2 has been in unstable since 2007-07-09. With help from Xavier Grave of Toy Lovelace fame, we now provide both the zero-cost and setjump/longjump exception handling mechanisms (i.e. two different versions of libgnat). ZCX is still the default and comes in both shared and static flavours. SJLJ is static only and is necessary for proper operation of the Distributed Systems annex. In addition to that, we are trying to add mips and mipsel to the list of supported architectures.

Xavier took over maintenance of the gnat-glade package and has had gnat-glade 2006 working with GCC 4.2 (SJLJ) experimentally since around November. However we decided not to upload it to unstable because...

... we decided to skip the transition to gnat-4.2 altogether, and go straight for gnat-4.3. As I reported earlier, upstream (and in particular Samuel Tardieu who deserves special thanks) has been very good at fixing bugs old and new. As a consequence, I feel that gnat-4.3 will be more stable and correct than gnat-4.2 (which is already pretty good). In addition, it seems suitable for building the latest versions of AWS and ASIS (but I'll report separately on that when the time comes).

Xavier and I ported all our patches from gnat-4.2 to gnat-4.3. I uploaded a prerelease of gnat-4.3 last week; it is now waiting for approval in the queue of new packages. Xavier is already working on porting gnat-glade to this prerelease.

The next steps are to stabilise gnat-4.3 to the point where we can make it the default Ada compiler and then start upgrading the other packages, starting, as usual, with asis and gnat-glade.

That's a lot of work and, as always, help is more than welcome.

[See also "Debian transition to GCC 4.2" in AUJ 28-4 (Dec 2007) —su]

Debian packaging

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 26 Feb 2008 05:51:02 −0800 (PST)
Subject: Re: Attn: Ludovic, on Debian Ada
Newsgroups: comp.lang.ada

> I have thought for some time that I would like to add my binding player-ada to Debian, and I think this will take some degree of implication on my part. It would be great if you could send me some pointers on what would be needed and how to get started. [...]

Great idea. As I have said many times, I will help anyone interested in packaging Ada things in Debian and even sponsor your packages. We'll take the details off-line, of course, but here are pointers to get you started.

As a prerequisite, you need a GPG or PGP key[1] signed by at least one Debian developer[2]. I suggest this as your first step, as setting up a physical meeting with a DD might take some time. In the mean time, of course, don't let that stop you from starting your packaging.

[1] GPG mini-HOWTO: http://www.dewinter.com/gnupg_howto/english/GPGMiniHowto.html

[2] Debian Key Signing Coordination page: https://nm.debian.org/gpg.php

Then dive into the technicalities:

[3] Debian New Maintainer's Guide: http://www.debian.org/doc/maint-guide/

[4] Work-Needing and Prospective Packages: http://www.debian.org/devel/wnpp/

[5] Debian Policy for Ada: http://www.ada-france.org/debian/debian-ada-policy.html

[6] Packaging scripts for all of my packages: http://www.ada-france.org/article131.html

OpenToken and TextTools are very simple packages you may want to look at for inspiration.

[See also "Official Debian Developers" in AUJ 27-3 (Sep 2006), p.43. —su]

Gentoo — Multiple compilers

From: George Shapovalov <george@gentoo.org>
Newsgroups: comp.lang.ada
Subject: Small changes in Ada handling in Gentoo
Date: Thu, 24 Jan 2008 07:39:30 −0800 (PST)

I have finally implemented the multiple gnat handling in full, resolving bug -151343. The operational procedures on user side remain largely the same, however there is one modification everybody using Ada compilers in Gentoo should be aware of.

Now (as was discussed in -151343) gnatbuild.eclass, gnat.eclass and eselect-gnat support the notion of "primary" compilers. The idea here is that most users will want one or, possibly, two variants of Ada compilers used for everyday work at most, while, occasionally, wishing to test some other variant(s).

These "everyday" compilers (designated "primary" further on) have to be listed in /etc/ada/primary-compilers. The file simply contains a listing of gnat profiles user desires to designate primary, one per line. Mine, for example, contains:

```
$ cat /etc/ada/primary_compilers
x86_64-pc-linux-gnu-gnat-gcc-4.2
x86_64-pc-linux-gnu-gnat-gpl-4.1
```

All Ada libs will be built only for these compilers and not for any other installed. However "eselect gnat set" will allow user to set any installed gnat, as before. Thus we avoid ABI issues when updating packages that have some Ada lib as a dependency (there are already a few) via enforcing a consistent set of ABIs, cut compilation time (libs are now not getting built for an "occasional" compiler) while still providing an ability to use any gnat in portage.

As a safeguard gnatbuild.eclass has also been modified to populate /etc/ada/primary-compilers, if this file does not exist yet, with the corresponding profile upon emerging gnat for the first time. Please, however, take your destiny in your own hands and update that file accordingly to your desires.

[See also "GNAT Split in Gentoo" in AUJ 27-4 (Dec 2006) —su]

References to Publications

"Computer Science Education: Where Are the Software Engineers of Tomorrow?" — CrossTalk

From: AdaCore Developer Center
Date: Monday January 7, 2008
Subject: Interesting article on the state of CS education
RSS: http://www.adacore.com/2008/01/07/interesting-article-on-the-state-of-cs-education/

Robert Dewar and Ed Schonberg recently published an article in CrossTalk discussing:

"Computer Science Education: Where Are the Software Engineers of Tomorrow?"

Abstract:

"It is our view that Computer Science (CS) education is neglecting basic skills, in particular in the areas of programming and formal methods. We consider that the general adoption of Java as a first programming language is in part responsible for this decline. We examine briefly the set of programming skills that should be part of every software professional's repertoire."

From: Slashdot
Subject: Professors Slam Java As "Damaging" To Students
Date: Tue Jan 08, 2008 03:18 AM
RSS: http://slashdot.org/article.pl?sid=08/01/08/0348239

jfmiller call to our attention two professors emeritus of computer science at New York University who have penned an article titled "Computer Science Education: Where Are the Software Engineers of Tomorrow?" [1] in which they berate their university, and others, for not teaching solid languages like C, C++, Lisp, and Ada. The submitter wonders whether any CS students or professors would care to respond. Quoting the article:

"The resulting set of skills [from today's educational practices] is insufficient for today's software industry (in particular for safety and security purposes) and, unfortunately, matches well what the outsourcing industry can offer. We are training easily replaceable professionals... Java programming courses did not prepare our students for the first course in systems, much less for more advanced ones. Students found it hard to write programs that did not have a graphic interface, had no feeling for the relationship between the source program and what the hardware would actually do, and (most damaging) did not understand the semantics of pointers at all, which made the use of C in systems programming very challenging."

[1] http://www.stsc.hill.af.mil/CrossTalk/2008/01/0801DewarSchonberg.html

From: Slashdot
Subject: Followup On Java As "Damaging" To Students
Date: Mon Jan 21, 2008 09:52 PM
RSS: http://slashdot.org/article.pl?sid=08/01/22/0217200

A prior article on the damage Java does to CS education was discussed here recently. There was substantial feedback and the mailbox of one of the authors, Prof Dewar, also has been filled with mainly positive responses. In this followup to the article, Prof. Dewar clarifies his position on Java [1]. In his view the core of the problem is universities 'dumbing down programs, hoping to make them more accessible and popular. Aspects of curriculum that are too demanding, or perceived as tedious, are downplayed in favor of simplified material that attracts a larger enrollment.'

[1] http://itmanagement.earthweb.com/career/article.php/3722876

From: ahab <ahabeger@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: Ada supportive artice discussed on /.
Date: Tue, 8 Jan 2008 10:54:09 −0800 (PST)

Same article made it onto http://programming.reddit.com/ , a few comments here http://programming.reddit.com/info/64utw/comments/

Reddit is very Python / Ruby / Scala centric.

Joachim Schueth Interviews

From: Frederik Sausmikat <frederik.sausmikat@gmx.de>
Date: Thu, 24 Jan 2008 21:52:50 +0100
Subject: [FYI] Joachim Schueth Interviews
Newsgroups: comp.lang.ada

As most of you already know, in November 2007 Joachim Schueth has beaten a reconstruction of the Colossus code breaking machine using the Ada programming language and a laptop running NetBSD.

Two interviews with Jo have been published, which might be of interest to you:

AdaCore: <http://www2.adacore.com/home/ada_answers/lorenz-code>

NetBSD: <http://www.netbsd.org/gallery/schueth-interview.html>

[See also "Ada helps win Cryptography Challenge" in AUJ 28-4 (Dec 2007) —su]

"There's nothing new about multicore mania" — EE Times

From: AdaCore Press Center
Date: Wednesday February 27, 2008
Subject: There's nothing new about multicore mania
RSS: http://www.adacore.com/2008/02/27/theres-nothing-new-about-multicore-mania/

The recent introduction of multicore architectures has caused a surprising amount of uproar. Multiprocessing has been around for decades. But it took the introduction of multicore chips by Intel and other manufacturers to bring multiprocessing to the attention of the public. And people are shocked to find out that many mainstream languages, including C and C++, are entirely ill-equipped for the task.

"Use Ada For Better Safety, Security, And Reliability" — Electronic Design

From: Ada Resource Association
Date: February 15, 2008
Subject: Electronic Design article
URL: http://adaic.com/whatsnew.html

Electronic Design publishes Use Ada For Better Safety, Security, And Reliability.

"Letters to the Editor: Ada's Tried and Tested" — SD Times

From: SD Times
Date: October 1, 2007
Subject: Letters to the Editor: Ada's Tried and Tested

URL: http://www.sdtimes.com/content/article.aspx?ArticleID=31209

The article "When Failure Isn't an Option," [Aug. 15, page 26] requires some clarification. It states that AdaCore is a member of the JSR 302 ("Safety Critical Java Technology") Expert Group. This is not accurate. A member of AdaCore's technical staff, Ben Brosgol, is an individual member of that expert group, but he is not there representing AdaCore.

One reason that AdaCore is not a corporate member of the JSR 302 group is that we have seen no interest in safety-critical Java from our customers. We produce development environments for safety-critical Ada systems, and our customers are much more interested in using tried and tested technology for a language—Ada—that was designed precisely for these sorts of applications, than to take the risk of moving to a language that intrinsically introduces major complications into the certification process.

We understand that some organizations make technology decisions based on what's popular vs. what's technically more fit to purpose, and that was the essence of my quote: "Language choice has always been significantly a matter of personal taste and enthusiasm, and there are lots of Java enthusiasts around." Unfortunately the quote was positioned so as to make it look like I agreed with that rationale.

As I explained in some other material that I furnished to the author, the real issue for managers of safety-critical projects is not which specific language the staff is familiar with—a competent programmer in any modern programming language should be able to learn a new language in short order. Rather, the more significant (and much harder to find) talent is the ability to develop large, safe systems, and that skill is rarely taught in universities. Java brings no advantages here.

Safety-critical Java is attracting a lot of "buzz" these days, in part because the technical issues that it raises tend to draw researchers who like to solve hard problems. But it is still very much a work in progress, as the article notes, and frankly a much riskier choice than Ada for a community that rightfully prides itself on conservatism. From our vantage point, Java seems a language of chance, not a language of choice, for safety-critical applications.

Robert Dewar

President and CEO

AdaCore

AdaCore — "Providing Effective Support for Software"

From: AdaCore Developer Center
Date: Wednesday January 16, 2008
Subject: Providing Effective Support for Software
RSS: http://www.adacore.com/2008/01/16/providing-effective-support-for-software/

In this paper, Emmanuel Briot and Robert Dewar discuss and describe an infrastructure for providing effective support for complex software tools. The purchase of software usually includes provisions for ongoing support, and indeed there are some fundamental reasons why software is a different kettle of fish when it comes to providing this support.

[http://www.adacore.com/wp-content/uploads/2008/01/effective_software_support.pdf —su]

AdaCore at Ada-Europe 2008

From: AdaCore Developer Center
Date: Monday February 4, 2008
Subject: Ada Europe 2008, 13th International Conference on Reliable Software Technologies
RSS: http://www.adacore.com/2008/02/04/ada-europe-2008-13th-international-conference-on-reliable-software-technologies/

AdaCore will present a number of papers, a tutorial, and will chair a discussion panel. AdaCore will also be exhibiting at this event.

Papers:

"A Comparison of the Object-Oriented Features of Ada 2005 and Java" — Ben Brosgol

"A Type Safe Database Interface" — Emmanuel Briot and Florian Villoing

"Exceptionally Safe" — Arnaud Charlet, Cyrille Comar, and Franco Gasperoni

Tutorials:

"Languages for Safety-Critical Software: Issues and Assessment" — Ben Brosgol

Panels:

"A Rational Approach to Software Engineering Education or: Java considered Harmful" — Ed Schonberg

AdaCore — EclipseCon 2008

From: AdaCore Press Center
Date: Wednesday December 19, 2007
Subject: EclipseCon 2008
RSS: http://www.adacore.com/2007/12/19/eclipsecon-2008/

Quentin Ochem will be giving a talk on "Bringing Ada to the Eclipse plug-in developers".

Wind River EMEA Aerospace and Defence Seminars

From: AdaCore Press Center
Date: Wednesday January 9, 2008
Subject: Wind River Regional Developer Conference Aerospace & Defense Edition
RSS: http://www.adacore.com/2008/01/09/wind-river-regional-developer-conference-aerospace-defense-edition-2/

AdaCore will be exhibiting at all of the RDC events. Dates and location are listed below:

March 12, 2008
Hilton Santa Clara
4949 Great America Parkway
Santa Clara, CA 95054
Tel.: 408-330-0001

March 27, 2008
Turf Valley Resort
2700 Turf Valley Road
Ellicott City, MD 21042
Tel.: 410-423-0833

April 2, 2008
Four Seasons Resort & Club Dallas at Las Colinas
4150 North MacArthur Boulevard
Irving, TX 75038
Tel.: 972-717-0700

[Click here to register http://www.rtcgroup.com/windriveraandd event/ —su]

[See also same topic in AUJ 28-4 (Dec 2007) —su]

Ada Inside

Saab — Real-Time Data Distribution Service (DDS)

From: AdaCore Press Center
Date: Saturday December 8, 2007
Subject: Working With Saab, AdaCore and RTI Integrate Support for Ada with DDS-Compliant Real-Time Messaging Middleware
RSS: http://www.adacore.com/2007/12/08/working-with-saab-adacore-and-rti-integrates-support-for-ada-with-dds-compliant-real-time-messaging-middleware/

Industry's first Ada bindings for AdaCore's GNAT Pro Compiler for development of high-performance distributed real-time applications.

SANTA CLARA, CA and NEW YORK, NY., November 12, 2007 — Real-Time Innovations (RTI), The Real-Time

Middleware Experts, today announced that it has integrated RTI Data

Distribution Service with an industry-leading Ada compiler, GNAT Pro from AdaCore Inc. Working closely with software engineers at Saab Systems, RTI has developed the first Ada bindings to support middleware compliant with the Data Distribution Service (DDS) for Real-Time Systems standard. For the first time, software developers can combine the unsurpassed messaging performance of RTI middleware, the portability and interoperability provided by the DDS standard, and the powerful development environment of AdaCore's GNAT Pro to build high-performance, fully standards-compliant distributed applications.

"RTI middleware with Ada integration is helping our developers build complex applications that require real-time data availability and response across large distributed systems," said Thomas Jungefeldt, senior systems engineer, Saab Systems, Naval Systems Division. "A major advantage of this approach is our ability to support and develop applications in a heterogeneous COTS-based environment requiring simple and straightforward integration of legacy code with newly developed systems."

"Adoption of the DDS standard is growing across a wide range of real-time distributed environments from desktop to embedded devices, particularly in defense and aerospace applications," commented Thomas Quinot, middleware specialist, AdaCore. "The integration of GNAT Pro with RTI's industry-leading real-time middleware is a critical part of our ongoing commitment to make Ada a development language of choice in high-performance distributed applications, allowing users to benefit from the strengths of both working together."

"The demand for DDS support from the Ada community is continuing to grow," explained David Barnett, vice president of Product Management at RTI. "AdaCore's GNAT Pro is available on more platforms than any other Ada technology, and we are excited to be the first to allow distributed application developers to take advantage of Ada technology in conjunction with RTI Data Distribution Service and the DDS standard."

About RTI Data Distribution Service

RTI Data Distribution Service is a high-performance messaging and data-caching solution for the development and integration of applications that require low latency, high throughput, high scalability, deterministic responses and minimal consumption of network, processor and memory resources. RTI Data Distribution Service is an open-architecture platform that complies with the Object Management Group's (OMG's) DDS for Real-Time Systems standard.

About Saab Systems

Saab Systems offers integrated command and control system solutions and civil security solutions, along with further development and adaptations of existing command and control systems. Saab Systems is a business unit within the Saab group and has around 1,200 employees in Australia, Denmark, Finland, South Africa and Sweden.

About RTI

Real-Time Innovations (RTI) provides high-performance infrastructure solutions for the development, deployment and integration of real time, data-driven applications. RTI's messaging, caching, Complex Event Processing (CEP) and visualization capabilities deliver dramatic improvements in latency, throughput and scalability while slashing cost of ownership. The company's software and design expertise have been leveraged in a broad range of industries including defense, intelligence, simulation, industrial control, transportation, finance, medical and communications. Founded in 1991, RTI is privately held and headquartered in Santa Clara, CA. For more information, please visit www.rti.com.

Indirect Information on Ada Usage

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

Job Description

[...] One of our most aggressive projects is creating a web-based, multi-media contact center solution. This is soft real-time software being developed in Ada. To help us accomplish this goal, we are hiring experienced Ada software engineers to add to our existing staff.

Candidates must be motivated, with a demonstrated ability to create useful abstractions. Must be able to work both individually and as part of a team. Will be integrating with VoIP solutions, and creating cutting-edge web-based user interfaces. Some light system administration will also be required. Adherence to strict coding standards is expected.

Must be able to program for Linux operating systems. Willing to consider this as a full-time remote/telecommuting position. Some knowledge of XHTML/XML/Javascript/CSS and VoIP are preferred, but not required. So if you are a self-motivated developer, we look forward to you joining our energetic team. [...]

Job Description: UK

I have a client located in Hampshire, UK who is looking for Ada Software Engineers on a permanent basis. They will also consider candidates with little or no Ada experience willing to be trained and have Ada as part of their current skill set.

Salaries are in the region of £40,000 p.a.

[...] many different and diverse organisations [...] who require Ada experience on both a contract and permanent basis and we are therefore always keen to hear from people with this skill set. We also have vacancies for a client located in Surrey. The salary would be similar to that quoted above.

Job Description: USA

[...] contractor for NASA at the White Sands Complex near Las Cruces, New Mexico. Operating here are two functionally identical satellite ground terminals: the White Sands Ground Terminal Upgrade, and the second TDRSS Ground Terminal. These two terminals ensure uninterrupted communications between various ground stations, NASA's orbiting fleet of Tracking and Data Relay satellites, customer spacecraft (satellites), and the computer systems that support such spacecraft. The WSC also serves as an interface for distributing satellite data to control centers and scientists who then use the daily influx of data to expand our ever growing knowledge of the Earth and the universe.

The Software Engineering Department at the White Sands Complex has openings junior through senior developers within several of our groups. [...]

Basic Qualifications:

- Software Engineer 1: BA/BS in mathematics, engineering, computer science or other related field and no experience or AA in related field and 4 years of related experience. In lieu of formal education, 8 years of related experience. With experience in: C++ on Linux, TCP/IP, MySQL, Unix/Linux scripting with Bash and Perl, Ada experience is an asset.

- Software Engineer 2: Minimum two years experience with a high level language (Ada preferred). MS in mathematics, engineering, computer science or other related field and no experience, BA/BS in related field and 2 years of related experience, or AA in related field and 6 years of related experience. In lieu of formal education, 10 years of related experience.

- Software Engineer 3: Minimum four years experience with a high level language (at least one year with Ada). MS in mathematics, engineering, computer science, or other related field 2 years of related experience, or BA/BS in related field and 4 years of progressive, related experience, or AA in related field and 8 years of progressive, related experience. In lieu of formal education, 12 years of progressive, related experience.

- Software Engineer 4: Minimum four years experience with multiple high level languages (at least two years with Ada required). MS in mathematics, engineering, computer science, or other related field and 4 years related experience, or BA/BS in related field and 8 years of progressive related experience, or AA in related field and 12 years of progressive related experience. In lieu of formal education, 16 years of progressive related experience.

- Software Engineer 5: We are looking for an experienced Ada developer that is self motivated and can quickly learn new applications. Minimum four years experience with multiple high level languages, no less than 2 years with Ada required. MS in mathematics, engineering, computer science, or other related field and 12 years related experience, or BA/BS in related field and 14 years of progressive related experience.

- Software Engineer 5 — TT&C Lead: Minimum 8 years experience with a high level language (at least four years of Ada). MS in mathematics, engineering, computer science or other related field and 12 years of experience, BA/BS in related field and 14 years of related experience. Minimum two years experience managing medium sized groups (6-10 people).

- Ability to obtain Secret security clearance. Must be US Citizen.

Job Description: USA

[...] small 8(a) company based in Maryland. We have more than a decade of experience with custom programming and database development. Our current focus has been on data visualization, modeling and simulation. We are currently in the DoD Mentor/Protégé program with the Navy.

[...] system where by an artificial signal is projected for reception by a radar system. This allows testing of the radar, missile systems, and personnel without having to physically perform a scenario or alter the radar system. Additional features will be added to this system in addition to porting the original system from Ada to C++, and SGI/Irix to possibly IBM Blade Server/Linux.

Programmer requirements:

Minimum: 3–4 yrs experience, Ada, C++, clearable to DoD Secret

Preferred:

5–10 yrs experience, Ada, C++, embedded programming, multi-threading, multi-processor, parallel processing, real-time systems, radar experience, Unix/Linux experience, DoD Secret clearance [...]

Ada in Context

Is the Ada Grammar context free?

From: Hibou57 <yannick_duchene@yahoo.fr>
Date: Tue, 19 Feb 2008 16:47:52 -0800 (PST)
Subject: Is it really Ok to assert that the Ada syntax is a context-free grammar ?
Newsgroups: comp.lang.ada

I got a doubt about [ARM 1.1.4-1]:

“The form of an Ada program is described by means of a context-free syntax together with context-dependent requirements expressed by narrative rules.”

But [...] as an example X(Y) can stand for a type cast, a function call, an array access, or even an array slice, and this cannot be decided without knowlegde of the context.

From: Robert A Duff <duff@adacore.com>
Date: Wed, 20 Feb 2008 15:51:13 −0500
Subject: Re: Is it really Ok to assert that the Ada syntax is a context-free grammar ?
Newsgroups: comp.lang.ada

[...] The grammar given in the Ada RM under “Syntax” is a context free grammar. It is, however, ambiguous, and therefore not LR(1).

You noted the biggest ambiguity — X(Y) could mean various things. There are a few others (e.g. .all can be implicit in some cases, and subprogram calls with no parameters don't get empty parens). The full power of semantic analysis is required to disambiguate these things, in the general case.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Wed, 20 Feb 2008 17:19:25 +0200
Subject: Re: Is it really Ok to assert that the Ada syntax is a context-free grammar ?
Newsgroups: comp.lang.ada

> [...] If it need name resolution to figure out what means the construct, this is not a context-free grammar.

The *grammar* in the ARM is surely context-free. In each production, there is only one non-terminal symbol on the left-hand side, for example 6.4(3):

function_call ::= prefix actual_parameter_part

In a context-dependent grammar, there would be several symbols on the left-hand side, before the “::=” sign.

The *Ada language* is not context-free, in the sense that the set of legal Ada programs cannot be defined by a context-free grammar. The same is true for all practical programming languages that allow user-defined identifiers with different properties, for example type identifiers and procedure identifiers, and that allow only some kinds of identifiers in some constructs. For example, you cannot “call” a type, but you can “call” a procedure.

The grammar in the ARM generates or accepts many strings that are not legal Ada programs. The “context-dependent narrative rules” are there to say which of these strings are legal Ada programs. This same definition method is used for most programming languages.

> Is this construct the only one of the grammar which is not context-free, or is there some others ?

Your are using the term “context-free” in some peculiar way, not in its usual meaning, I think.

It seems to me that you are really asking this question: given a *fragment* of an Ada program, for example “X(Y)”, can this fragment be produced from more than one non-terminal grammar symbol?

I don't think that this has much to do with the context-freeness of the grammar, and I don't understand why you feel so strongly that there should be only one grammar symbol for each fragment.

For another example, consider the fragment

```
delay 0.3
```

The way the grammar is now written, this fragment can (I think) be produced only from the non-terminal symbol delay_statement, by the rule in ARM 9.6(4). Does that make you happy? But look, the *meaning* of this fragment depends strongly on the context of this delay_statement: is it used as a

simple_statement (5.1(4))

or a delay_alternative (9.7.1(6))

or a triggering_statement (9.7.4(4)) ?

These grammar rules could also be written without using the non-terminal symbol delay_statement, replacing it by the sequence consisting of the terminal symbol “delay” and the non-terminal symbol “expression”. Then the fragment “delay 0.3” could be produced directly by any one of the three different non-terminal symbols (simple_statement, delay_alternative, triggering_statement). I don't think that change in the grammar would make any significant difference to the readability or parsability of Ada, so I don't see what your problem is.

A Coding War Story

From: Kickin' the Darkness
Date: Thursday, December 6, 2007
Subject: A Coding War Story: What's Your Point?
RSS: http://blog.kickin-the-darkness.com/2007/12/coding-war-story-whats-your-point.html

I had been assigned the task of porting a fairly large (about 400 KSLOC) missile launch command and control system to an upgraded OS version and new compiler and language version. Specifically, from Solaris 2.5.1 to Solaris 7, and from the Verdix Ada Development System (VADS), which was Ada 83, to Rational Apex Ada, which was Ada 95. VADS had been bought out by Rational, and its product obsoleted, although Rational did a pretty good job implementing compatible versions of VADS-specific packages to ease the transition to the Apex compiler.

Three other guys helped with the initial compilations, just to get clean compiles of the code, which took about two weeks, and then I was on my own to actually make the whole system work. Long story short, it was the worst design and implementation of a software system I'd ever seen, and so took about two more months to successfully complete the port. It was then handed over for formal testing, which took several months as well. I fairly steadily fixed the bugs that were found as testing got going, but that rate quickly declined as it progressed (the original code was a production system after all, so its functionality was pretty solid, I just had to kill the bugs that came about due to adapting to the new compiler). Eventually I was reassigned to another project once everything appeared to be working as well as the original.

Then came the phone call on the Friday before Thanksgiving.

There was a missile test scheduled in about three weeks, and during a lab countdown test the command sequencing had locked up. (...) Like most safety-critical defense systems like this, a lot of logging is captured, so it was fairly easy to locate the handful of lines of code that had been most recently executed when the system froze. And of course there was absolutely nothing questionable in those lines of code, and these same statements had already successfully executed literally thousands of times during that same run.

We put the Apex guys at Rational on notice, since it was their compiler and some of their vendor-supplied routines were being called in this area, and it was impressed on them (and everyone) that this was a problem of literally national importance that had to be tracked down. So they got their Thanksgiving week trashed as well.

Since the logs could only tell us so much, we needed to try to repeat the problem in the local lab. For something that pops up in only 1 in a 1000 test runs that's not going to be easy. Amongst the conjectures as to root cause was that a call into a vendor-supplied mutex (part of a VADS migration package) Unlock function was not unlocking. (...) About an hour later the system locked up. (...)

The trick now was to figure out exactly where in the sequence of candidate statements the lock up was occurring.

The implementation of this system used Ada tasking, and used it extraordinarily poorly. (...) Once rendezvous had been made with a target task, that target task would then rendezvous with another task, which in turn would rendezvous with another task, and so on, until eventually some processing would get done, after which all the rendezvous would be broken and each of the tasks would go on their merry way. So what you ended up with was the world's most expensive function calls, bringing an entire, “multi-tasking” process to a halt while it processed a piece of incoming data. It was only because the normal throughput was so low that this hadn't caused performance problems in the past. (...)

So in tracking down exactly which line was causing the problem I had to find a way to record the progress through the sequence of statements (...) The process as a whole wasn't getting blocked, just a (critical) task chain within it.

This was the wedge needed to get at locating the offending statement.

I created an Ada package containing an enumeration type, a global variable of that type, and a task. (...) The monitoring task then itself did nothing more than loop and periodically check to see if the global variable had changed value. Every time it did, it printed out the value to a file. It then delayed for a small interval, and made its next check. Now the reason I could write to a file from this task was that this task only ran when a task switch had occurred back in the problem area and this task had been selected to run. Whatever was done in this task should have no effect on other, unrelated, blocked tasks. (...)

Ran the instrumented executable. It froze up. And the monitoring worked like a charm.

The logging of the progress monitoring variable displayed exactly the anticipated sequence, which eventually ceased with a value corresponding to having made a call to the Mutex Unlock function, with the value that should have been stored signaling the resumption of the task never showing up—like it had in the thousands of previous invocations.

So over to you Rational. The Apex engineers during this time had been feverishly analyzing their code and had found a place in the mutex code where it could theoretically block for good, but the odds of that happening were very remote because of everything that had to happen with the right sequencing and timing. Murphy's Law, guys, Murphy's Law.

What I did to work-around this was to replace the calls to the vendor's mutex functions (which were built atop the OS' mutex functionality) protecting this particular sequence of code with a quick little native Ada mutex package, using that to control mutex access to the relevant area.

I put this into the code and reran the test. Seven hours later it was still running.

My mutex package code was given to Rational who compiled and disassembled it and verified that it was not using the same approach that the problematic mutex functions were using.

I then had the most well attended code inspection of my career :-) There were nearly a dozen engineers and managers in the room with me, and at least another dozen dialed in from all over the country, all to inspect about 20 lines of code.

It passed, the new executables were formally built, and it was handed over to the test organization for formal regression testing. A couple weeks later the missile countdown proceeded flawlessly and away it went.

It's a good think I like cold turkey.

Okay, this is all well and fine, but what's really the point of a coding war story?

This was a nasty, nasty problem. There was concurrency, over a dozen communicating processes, hundreds of KSLOCs, poor design, poor implementation, interfaces to embedded systems, and millions of dollars riding on the effort. No pressure, eh?

I wasn't the only developer working on this problem, though having done the original port I was of course the primary focus. But even though I did the porting, that doesn't mean I had intimate knowledge of hundreds of thousands of lines of code—or even a decent overview of it. Other engineers around the country were looking through the code and the logs as well, but I found that when they proposed a hypothesis to me about a root cause, it never took more than 30 seconds on my part to dismiss it, likewise when I was requested to provide various analyses I would shove it off on to someone else because it was clear to me they were on the wrong track. Sound like arrogance on my part? Well, yeah, it does, but that's not why I dismissed these hypotheses and requests.

It was because I knew what the nature of the problem was. I didn't know exactly where it was occurring, nor why it was occurring, but I did know what was happening.

I've built up a lot of experience and knowledge over the years—I was an early adopter of Ada, understand concurrency and its pitfalls, I know how Ada runtime libraries handle tasking and concurrency, and I understand low-level programming at the level of raw memory, registers, and assembly language. In other words, I have deep knowledge of my niche of the industry. All of that was brought to bear in successfully tracking down this problem—not just working around the bug, but understanding how to put together an approach to finding the bug in a very sensitive execution environment.

The specifics of a coding war story probably aren't all that interesting to those who aren't familiar with the particulars of its nature and environment, but they are useful for gleaning an understanding of what it takes to solve really difficult problems.

To solve the really difficult problems you need to be more than a coder, you have to understand the "fate" of that code, how it interacts with its environment, and how its environment itself operates.

Then you too can get your Thanksgiving holiday all messed up.

Marc A. Criley

Command line in DOS

From: Charles H. Sampson <csampson@inetworld.net>
Date: Tue, 4 Dec 2007 19:34:26 −0800
Subject: DOS Options
Newsgroups: comp.lang.ada

I've been using GNAT 3.15p for a long time to write DOS-like project utilities to run on my Wintel desktop. I've just come up with a case where I'd like to implement optional parameters. Since the utility is DOS-like, I want to use the DOS form of optional parameters: attached to the name of the command, separated by the '/' character. [...]

From: Jeffrey Creem <jeff@thecreems.com>
Newsgroups: comp.lang.ada
Subject: Re: DOS Options
Date: Wed, 05 Dec 2007 05:15:01 GMT

Ada.Command_Line.Argument(1) certainly returns the first argument to the program regardless of whether it is DOS, Unix, etc. It even appears to handle the degenerate case of no space between the command and the argument. [...] Since

```
my_command /option_1 /option_2
```

and even

```
my_command/option_1 /option_2
```

work such that Ada.Command_Line.Argument(1) returns /option_1 and Ada.Command_Line.Argument(2) returns /option_2

[...]

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Wed, 19 Dec 2007 20:40:36 +0100
Subject: Re: DOS Options
Newsgroups: comp.lang.ada

> [...] I found out that, for the GNAT 3.15p version of Ada.Command_Line, if the command name is followed immediately by a '/', that slash and everything following it up to the first blank is presented as Argument (1). From that point on, blanks act as separators between parameters.
>
> This has the effect that it's still necessary to write a parser for Argument (1), something I thought Ada.Command_Line might be taking care of. I'm talking about the case of multiple slash-headed options jammed against the command:
>
> command/x/y/z
>
> I was hoping that Ada.Command_Line would return "/x" as Argument (1), "/y" as Argument (2), etc.

[...] ACT doesn't support DOS since version 3.07p or maybe even before. The last on-purpose packaged version of GNAT (specifically as a GNAT Public version, not a GCC version) for DOS is 3.10p.

Your 3.15p is certainly the Windows version, and what you are seeing is a Win32 console output. Indeed, you need to congratulate Microsoft which decided to parse and separate the arguments in Win32 in a fashion consistent with DOS — adding the parsing of filenames containing spaces, enclosed by "". Ada.Command_Line.Argument (i) just gives the "ith" argument from the system, as the system wants to give it, that's it. Running the following Turbo Pascal (authentic DOS) code:

```
VAR i:Word;BEGIN FOR i:=1 TO
paramcount DO
WriteLn(i,':',paramstr(i))END.
```

you get exactly the same (expected, even if not ideal) behaviour:

```
C:\TEMP>show_cmd/x/y/z /a/b/c
1:/x/y/z
2:/a/b/c
```

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: DOS Options
Date: Wed, 19 Dec 2007 16:47:14 −0600

Ada.Command_Line is a thinly-veiled wrapper on the normal C "argv/argc" mechanism. I think most Ada compilers simply copy that mechanism (that's pretty much a requirement on Unix, and it is better for portability on all platforms). The fact that that mechanism isn't very useful is not considered relevant. (And yes, for the record, Janus/Ada does exactly that for Ada.Command_Line. We don't even bother to use Ada.Command_Line for the command line processing of our compiler and tools; it is just as easy to process starting from the original string.)

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: DOS Options
Date: Wed, 19 Dec 2007 16:59:25 −0600

[...] Both MS-DOS and Win32 only provide a single string as the result of the command line — the system provides no parsing whatsoever. For Win32, the function is GetCommandLine. In DOS, you had to grab the string from the memory of your executable process — there isn't even a function to get it.

Parsing is something that was provided by C compilers on both of those platforms. The was generally defined to be compatible with the Unix C implementations (since that is how C was originally defined). Janus/Ada never did any parsing at all (just returning a single string, even on Unix); if we had done some, it would have separated the options out as the original OP suggested. But that would be fairly complex to do, we never figured out a good way to encapsulate it for our products, so we didn't make that library available to our customers.

I'm a little stunned that Turbo Pascal would have copied this useless parsing definition, given that they had to write all of the code themselves. But perhaps it is the same reason that we ended up doing so for Ada.Command_Line: to be compatible with C compilers and/or (possible) Unix versions of your product.

In any event, Microsoft's OSes have had nothing whatsoever to do the the parsing of command lines. All of the code to parse the command line appears inside of Ada.Command_Line for a Windows implementation (or perhaps some lower-level library intended to emulate Unix behavior).

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: DOS Options
Date: Thu, 20 Dec 2007 16:15:42 −0600

> Windows does appear to have a CommandLineToArgvW function that parses the command line (or whatever string you give it). I don't know whether GNAT uses it. I'm pretty sure that not all C compilers use it—the MINGW32 version, I believe, does its own parsing, in whatever startup routine is linked into the program before main() is called.

Fascinating; never noticed that one. I doubt anything much uses it, though, because it isn't supported on Windows 95/98/ME, so any compiler runtime that used it would have to be almost brand-new. (It would have been nasty to make programs that don't run on older Windows systems just to use a function that is easy to write yourself. Not as big of an issue today, but I doubt compiler vendors are rewriting their runtime to purposely break something that previously worked!)

Fixing bugs in GCC

From: Chris <echancrure@gmail.com>
Date: Mon, 25 Feb 2008 13:55:38 −0800 (PST)
Subject: bug report being ignored
Newsgroups: comp.lang.ada

[...] I am just ignorant of the gcc/ gnat process/effort involved. I filled a bug report for gnat on bugzilla [...] and so far it has remained unconfirmed, never mind looked into in view to be fixed.

It's not always easy to ask question like this without offending gcc/ gnat people who are doing a tremendous job.

It is an annoying bug for me as it has an impact on gnatxref that I use a lot.

Is there anything that I can do to try to get that bug fixed?

From: Samuel Tardieu <sam@rfc1149.net>
Date: Mon, 25 Feb 2008 23:33:32 +0100
Subject: Re: bug report being ignored
Newsgroups: comp.lang.ada

As far as I'm concerned, I take no offense, but I will try to explain the current situation of bug reports against the FSF GNAT tree.

You're not doing anything wrong... except that you expect non-paid volunteers to do the job in a short time window. As one of the (very few) people who fix GNAT bugs in the FSF version, I choose the issues to address according to:

- the time I have (I do this on my free time, and I have a full-time job and many other computer-related and computer-unrelated activities);

- the tools I use (I've never needed gnatxref for my Ada development);

- the severity of the bugs according to my own ratings (an internal compiler error is high on the list, especially when it is triggered by valid real-world Ada code, a problem on a specially-crafted example concerning an obscure RM rule is low on my list).

The main issue is the first one I describe, i.e. the time to work on GNAT. A few months ago, I started fixing the bugs one after another, in the hope that more people would join the effort. Some of them (in particular Ludovic Brenta and Duncan Sands) do a tremendous job for the community, but we really lack manpower. AdaCore is also very helpful with volunteers working on the public version of GCC; we *need* to find more people to work on FSF GNAT.

So unless someone gets excited by your bug report, you would better try to fix it yourself if you have the needed expertise and time, or wait until it gets fixed by AdaCore because one of their clients has reported it.

The main point is: when people work on Free Software as a hobby, they will set their own agenda. And when there aren't many people, not many topics are addressed at once.

Anyway, thanks for your bug report. Properly reporting issues is already an important way to contribute back to Free Software you use.

Happy Birthday Ada Lovelace

From: John McCormick <mccormick@cs.uni.edu>
Date: Mon, 10 Dec 2007 13:10:10 −0800 (PST)
Subject: Happy Birthday Ada Lovelace
Newsgroups: comp.lang.ada

On December 10, 1815, Anna Isabella (Annabella) Byron, whose husband was Lord Byron, gave birth to a daughter, Augusta Ada. Ada's father was a romantic poet whose fame derived not only from his works but also from his wild and scandalous behavior. His marriage to Annabella was strained from the beginning, and Annabella left Byron just a little more than a month after Ada was born. By April of that year, Annabella and Byron signed separation papers, and Byron left England, never to return.

Byron's writings show that he greatly regretted that he was unable to see his daughter. In one poem, for example, he wrote of Ada,

> I see thee not.
>
> I hear thee not.
>
> But none can be so rapt in thee.

Byron died in Greece at the age of 36, and one of the last things he said was,

> Oh my poor dear child! My dear Ada! My God, could I but have seen her!

Meanwhile, Annabella, who was eventually to become a baroness in her own right, and who was herself educated as both a mathematician and a poet, carried on with Ada's upbringing and education. Annabella gave Ada her first instruction in mathematics, but it soon became clear that Ada's gift for the subject was such that it required more extensive tutoring. Ada received further training in mathematics from Augustus DeMorgan, who is today famous for one of the basic theorems of Boolean algebra, which forms the basis for modern computers. By the age of eight, Ada also had demonstrated an interest in mechanical devices and was building detailed model boats.

When she was 18, Ada visited the Mechanics Institute to hear Dr. Dionysius Lardner's lectures on the "difference engine," a mechanical calculating machine being built by Charles Babbage. She became so interested in the device that she arranged to be introduced to Babbage. It was said that, upon seeing Babbage's machine, Ada was the only person in the room to understand immediately how it worked and to appreciate its significance.

Ada and Babbage became good friends and she worked with him for the rest of her life, helping to document his designs, translating writings about his work, and developing programs to be used on his machines. Unfortunately, Babbage never completed construction of any of his designs. Even so, today Ada is recognized as being the first computer programmer in history. That title, however, does not do full justice to her genius.

Around the time that Babbage met Ada, he began the design for an even more ambitious machine called the "analytical engine," which we now recognize was the first programmable computer. Ada instantly grasped the implications of the device and foresaw its application in ways that even Babbage did not imagine. Ada believed that mathematics eventually would develop into a system of symbols that could be used to represent anything in the universe. From her notes, it is clear that Ada saw that the analytical engine could go beyond arithmetic computations and become a general manipulator of symbols, and thus it would be capable of almost anything. She even suggested that such a device could eventually be programmed with rules of harmony and composition so that it could produce "scientific" music. In effect, Ada foresaw the field of artificial intelligence over 150 years ago.

In 1842, Babbage went to Turin, Italy, and gave a series of lectures on his analytical engine. One of the attendees was Luigi Menabrea, who was so impressed that he wrote an account of Babbage's lectures. At age 27, Ada decided to translate the account into English, with the intent to add a few of her own notes about the machine. In the end, her notes were twice as long as the original material, and the document, "The Sketch of the Analytical Engine," became the definitive work on the subject.

It is obvious from Ada's letters that her "notes" were entirely her own and that Babbage was acting as a sometimes unappreciated editor. At one point, Ada wrote to him,

"I am much annoyed at your having altered my Note. You know I am always willing to make any required alterations myself, but that I cannot endure another person to meddle with my sentences."

Ada gained the title Countess of Lovelace when she married Lord William Lovelace. The couple had three children, but Ada was so consumed by her love of mathematics that she left their upbringing to her mother. For a woman of that day, such behavior was considered almost as scandalous as some of her father's exploits, but her husband was actually quite supportive of her work.

In 1852, Ada died from cancer. Sadly, if she had lived just one year longer, she would have witnessed the unveiling of a working difference engine built from one of Babbage's designs by George and Edward Scheutz in Sweden. Like her father, Ada lived only until she was 36, and, even though they led much different lives, she undoubtedly admired Byron and took inspiration from his unconventional and rebellious nature. At the end, Ada asked to be buried beside him at the family's estate.

Ada Lovelace biography material excerpted from "Programming and Problem Solving with Ada" by Dale, Weems, and McCormick. Jones & Bartlett Publishers, 2000.

The film "To Dream Tomorrow" from Flare Productions, www.flarefilms.org, tells the story of Ada Lovelace and her contributions to computing. I recommend it to all comp.lang.ada subscribers. See if you can find the error on her tomb.

Conference Calendar

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

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

2008

April 2	**ICECCS2008 - 3rd Workshop on UML and AADL** (UML&AADL'2008), Belfast, Northern Ireland, UK. Topics include: code generation from a high-level specification, architecture model verification, modeling Distributed Real-time Embedded systems with an MDA approach, etc.
April 09-11	2nd **International Conference on Tests And Proofs** (TAP'2008), Prato (near Florence), Italy.
April 14-17	21st **Conference on Software Engineering Education and Training** (CSEET'2008), Charleston, South Carolina, USA.
☺ April 14-18	22nd IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2008), Miami, Florida, USA. Topics include: all areas of parallel and distributed processing, such as Applications of parallel and distributed computing; Parallel and distributed software, including parallel programming languages and compilers, runtime systems, middleware, libraries, and programming environments and tools, etc.
	☺ April 14-18 9th **International Workshop on Parallel and Distributed Scientific and Engineering Computing** (PDSEC-08). Topics include: parallel and distributed computing techniques and codes, practical experiences using various parallel and distributed systems, task parallelism, compiler issues for scientific and engineering computing, applications, etc.
April 29 – May 02	**Systems and Software Technology Conference** (SSTC'2008), Las Vegas, Nevada, USA.
☺ May 05-07	11th IEEE **International Symposium on Object/component/service-oriented Real-time distributed Computing** (ISORC'2008), Orlando, Florida, USA. 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.); ...
May 07-09	7th **European Dependable Computing Conference** (EDCC-7), Kaunas, Lithuania. Topics include: Architectures for dependable systems; Fault tolerant distributed systems; Fault tolerance in real-time systems; Hardware and software testing, verification, and validation; Formal methods for dependability; Safety-critical systems; Software reliability engineering; Software engineering for dependability; etc.
☺ May 10-18	30th **International Conference on Software Engineering** (ICSE'2008), Leipzig, Germany. Topics include: Software components and reuse, Theory and formal methods, Engineering secure software, Software dependability, safety and reliability, Reverse engineering and maintenance, Software economics and metrics, Empirical software engineering, Engineering of distributed/parallel software systems, Engineering of embedded and real-time software, Software tools and development environments, Programming languages, etc. Deadline for early registration: April 1, 2008.
	☺ May 11 **International Workshop on Multicore Software Engineering** (IWMSE'2008). Topics include: Modeling techniques for multicore software; Software components and composition; Programming languages/models for multicore software; Compilers for parallelism; Testing and debugging parallel applications; Software reengineering for parallelism; Operating system support, scheduling; Development environments for multicore software; Experience reports from research or industrial projects; etc. Deadline for early registration: April 1, 2008.

May 20	ICRA2008 - 3rd **Workshop on Software Development and Integration in Robotics** (SDIR-III), San Diego, CA, USA. Topics include: Analysis of issues and challenges in robotic software development; Architectural models that lead to reusable robotic software design; Middleware services and reusable components for real time robot software systems; Description of state-of-the art research projects, innovative ideas, field-based studies; Identifying real-time requirements for robotic applications; Comparing existing development approaches for real-time applications; etc.
May 25-29	10th **International Conference on Software Reuse** (ICSR'2008), Beijing, China. Topics include: Confidence Ensuring and Evaluating Methods; Processes to identify and select OTS components; Software integration and evolution problems; Software variability management; Software generators and domain-specific languages; Component-based software engineering; Evolution of component-based software systems; Lightweight approaches to software reuse; Benefit and risk analysis of reuse investments; Generation of non-code artifacts; Quality aspects of reuse, e.g. security and reliability; Success and failure stories of reuse approaches from industrial context; etc.
May 26-30	15th **International Symposium on Formal Methods** (FM'2008), Turku, Finland. Topics include: all aspects of formal methods research, both theoretical and practical, in particular the experience of applying formal methods in practice.
☺ May 27-30	**DAta Systems In Aerospace** (DASIA'2008), Palma de Majorca, Spain.
June 04-06	10th IFIP **International Conference on Formal Methods for Open Object-based Distributed Systems** (FMOODS'2008), Oslo, Norway. Topics include: Semantics and implementation of object-oriented programming and (visual) modelling languages; Formal techniques for specification, design, analysis, verification, validation and testing; Model checking, theorem proving and deductive verification; Model transformations and refactorings; Applications of formal methods; Experience report on best practices and tools; etc.
June 04-06	8th IFIP **International Conference on Distributed Applications and Interoperable Systems** (DAIS'2008), Oslo, Norway. Topics include: innovative distributed applications; models and concepts supporting distributed applications; middleware supporting distributed applications; software engineering of distributed applications; etc.
☺ June 07	**Programming Language Approaches to Concurrency and Communication-centric Software** (PLACES'2008), Oslo, Norway. Topics include: the general area of foundations of programming languages for concurrency and distribution, such as multicore and network-on-chip programming, integration of sequential and concurrent programming, program analysis, concurrent data types, etc.
☺ June 09-11	8th **International Conference on Algorithms and Architectures for Parallel Processing** (ICA3PP'2008), Cyprus. Topics include: Multi-core Programming and Software Tools, Parallel Programming Paradigms, Tools & Environments for Parallel & Distributed Software Development, etc.
June 09-12	4th **European Conference on Model Driven Architecture Foundations and Applications** (ECMDA'2008), Berlin, Germany. Topics include: Model Transformation - languages and tools; Reverse Engineering; 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; MDA for Legacy Systems; MDA and Component-Based Software Engineering; etc. Deadline for submissions: April 7, 2008 (tools and posters).
♦ June 16-20	13th **International Conference on Reliable Software Technologies – Ada-Europe 2008**, Venice, Italy. Organized and sponsored by Ada-Europe, in cooperation with ACM SIGAda. Deadline for early registration: May 31, 2008.
June 17-19	2nd IEEE **International Symposium on Theoretical Aspects of Software Engineering** (TASE'2008), Nanjing, China. Topics include: Specification and Validation, Component-based Development, Model Checking for Software, Software Architectures and Design, Software safety and reliability, Reverse Engineering and Software Maintenance, Embedded and Real-time Software, Model-driven Development, Parallel and Distributed Computing, Program Analysis, Semantics and Design of Programming Languages, etc.
June 17-20	28th **International Conference on Distributed Computing Systems** (ICDCS'2008), Beijing, China. Topics include: theoretical foundations, reliability and dependability, security, middleware, etc.

June 25-27 **Code Generation 2008**, Cambridge, UK. Topics include: Tool and technology adoption, Defining and implementing modelling languages, Language evolution and modularization, Runtime virtual machines versus direct code generation, etc.

☺ June 27 DSN2008 - **Workshop on Architecting Dependable Systems** (WADS'2008), Anchorage, Alaska, USA. Topics include: everything related to software architectures for dependable systems, such as: Rigorous design: architectural description languages, formal development, ...; Verification & validation: theorem proving, type checking, ...; Fault tolerance; System evaluation; Enabling technologies; Application areas: safety-critical systems, embedded systems, ...; etc.

June 30 – July 02 13th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2008), Madrid, Spain.

☺ June 30 – July 04 **Technology of Object-Oriented Languages and Systems** (TOOLS Europe'2008), Zurich, Switzerland. Topics include: all modern approaches to software development, with a special but not exclusive emphasis on O-O and components.

July 01 ECRTS'08 - 8th **International Workshop on Worst-Case Execution Time Analysis** (WCET'2008), Prague, Czech Republic. In conjunction with the 20th ECRTS conference. Topics include: any issue related to timing analysis, such as Tools for timing analysis, Design for timing predictability, Integration of WCET analysis into the development process, etc. Deadline for paper submissions: April 14, 2008.

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

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

> ☺ July 06 1st **Interaction and Concurrency Experience** (ICE'2008). Topics include: Synchronous and Asynchronous Interactions in Concurrent Distributed Systems; models, logic and types for interactions; synchronous/asynchronous mechanisms; expressiveness results; timed and hybrid interactions; verification, analysis and tools; programming primitives for interactions. Deadline for submissions: April 14, 2008 (abstracts), April 18, 2008 (papers).

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

☺ July 07-11 22nd **European Conference on Object Oriented Programming** (ECOOP'2008), Paphos, Cyprus. Topics include: analysis, design methods and design patterns; concurrent, real-time or parallel systems; distributed systems; language design and implementation; programming environments and tools; type systems, formal methods; compatibility, software evolution; components, modularity; etc. Deadline for early registration: June 1, 2008.

> ☺ July 07 18th **Doctoral Symposium and PhD Students Workshop**. Topics include: Design methods and design patterns; Concurrent, real-time or parallel systems; Distributed

systems; Language design and implementation; Programming environments and tools; Type systems, formal methods; Software evolution; Components, Modularity; etc. Deadline for submissions: May 1, 2008.

☺ July 07 7th **Workshop on Parallel/High-Performance Object-Oriented Scientific Computing** (POOSC'2006). Topics include: tried or proposed programming language alternatives to C++; issues specific to handling or abstracting parallelism, including the handling or abstraction of heterogeneous architectures; existing, developing, or proposed software; grand visions (of relevance); etc. Deadline for submissions: May 14, 2008.

☺ July 07 **International Workshop on Advanced Software Development Tools and Techniques** (WASDeTT'2008). Topics include: What features in object-oriented languages make them easier to build tools for/with? Deadline for submissions: April 30, 2008.

☺ July 07 **International Workshop on Object-Oriented Software Development for the Embedded World** (OOSDEW'2008). Topics include: object-oriented language features for embedded devices; software techniques and tools for optimizing code for embedded devices; etc. Deadline for submissions: May 4, 2008.

☺ July 07 3rd **Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems** (ICOOOLPS'2008). Topics include: Topics include: implementation of fundamental OOL features: inheritance (object layout, late binding, subtype test, ...), genericity (parametric types), memory management; runtime systems: compilers, linkers, etc; optimizations: static and dynamic analyses, threads and synchronization, etc; resource constraints: real-time systems, embedded systems; relevant choices and tradeoffs: separate compilation vs. global compilation, dynamic checking vs. proof-carrying code, annotations vs. no annotations, etc. Deadline for submissions: May 4, 2008.

July 07-13 20th **International Conference on Computer Aided Verification** (CAV'2008), Princeton, USA. Topics include: Algorithms and tools for verifying models and implementations, Program analysis and software verification, Modeling and specification formalisms, Applications and case studies, Verification in industrial practice, etc.

☺ July 07-08 **Workshop on Exploiting Concurrency Efficiently and Correctly** ((EC)^2). Topics include: advances in programming languages and tools for developing concurrent software; programming constructs for concurrency; formalization of concurrency libraries; verification tools; introducing concurrency in education; etc. Deadline for position paper submissions: April 10, 2008.

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

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

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

☺ July 20 **International Workshop on Defects in Large Software Systems** (DEFECTS'2008). Topics include: Techniques to detect, locate, or predict defects; Empirical studies of defects; Types of defects that occur in software; Evolution of defects over time; Tools for post-deployment defect detection and reporting; Experience using certain techniques to identify or predict defects; etc. Deadline for position paper submissions: April 10, 2008.

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

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

☺ August 26 EuroPar2008 - 2nd **Workshop on Highly Parallel Processing on a Chip** (HPPC'2008). Topics include: programming models; languages and software libraries; implementation techniques (e.g. multi-threading, work-stealing); support and performance tools, performance evaluation; parallel algorithms and applications; etc.; for/on highly parallel multi-core systems. Deadline for submissions: June 6, 2008.

☺ August 27-29 12th **Brazilian Symposium on Programming Languages** (SBLP'2008), Fortaleza, Ceara, Brazil. Topics include: Programming language design and implementation; Design and implementation of programming language environments; Object-oriented programming languages; New programming models; Program transformations; Program analysis and verification; Compilation and interpretation techniques; etc. Deadline for submissions: April 4, 2008 (papers).

☺ September 03-05 7th **International Conference on Distributed and Parallel Systems** (DAPSYS'2008), Debrecen, Hungary. Topics include: Distributed and Grid middleware, Parallel and distributed programming languages and algorithms, Formal models for parallel and distributed computing, Software engineering and development tools, etc. Deadline for paper submissions: April 1, 2008. Deadline for early registration: May 8, 2008.

☺ September 07-10 9th **Conference on Communicating Process Architectures** (CPA'2008), York, UK. Topics include: Theoretical approaches to concurrency, and formal languages supporting these approaches, including the integration of existing formal notations; Modelling of, and model-driven development of concurrent software architectures; Verification and analysis of concurrent systems; Model-checking techniques and tools for development and analysis; Tools and languages for hardware-software co-design; Programming languages and environments for concurrent systems; Programming and implementation issues for concurrent languages, such as deadlock-freedom by design, starvation, and efficient inter-process communication architectures; System issues for programming languages supporting concurrency, such as multithreading kernels and interrupt architectures; Applications that exploit, or rely on, concurrency; etc. Deadline for paper submissions: April 25, 2008. Deadline for early registration: June 30, 2008.

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

☺ October 06-08 27th IEEE **International Symposium on Reliable Distributed Systems** (SRDS'2008), Napoli, Italy. Topics include: High-confidence systems, Critical infrastructures, Distributed embedded systems, Formal methods and foundations for dependable distributed computing, etc. Deadline for submissions: April 20, 2008 (abstracts), April 28, 2008 (papers).

October 06-10 2nd IFIP **Working Conference on Verified Software: Theories, Tools, Experiments** (VSTTE'2008), Toronto, Canada. Topics include: all aspects of verified software, theoretical as well as experimental, such as specification languages and case-studies, programming languages, language semantics, software design methods, automatic code generation, type systems, verification tools (static analysis, dynamic analysis, model checking, theorem proving, satisfiability), integrated verification environments, etc. Deadline for submissions: April 30, 2008.

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

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

	telecommunications, process control, multimedia, inhouse entertainment, robotics); etc. Deadline for paper submissions: April 26, 2008.
☺ October 19-23	23rd **Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications** (OOPSLA'2008), Nashville, USA. Topics include: new and better programming and design paradigms as well as practices. Deadline for submissions: July 2, 2008 (Development Program Proposals, Student Research Competition, Onward! short papers and films, Doctoral Symposium and Student Volunteers).
☺ October 19-24	**Embedded Systems Week 2008** (ESWEEK'2008), Atlanta, Georgia, USA. Includes CASES'2008 (International Conference on Compilers, Architecture, and Synthesis for Embedded Systems), CODES+ISSS'2008 (International Conference on Hardware/Software Codesign and System Synthesis), EMSOFT'2008 (International Conference on Embedded Software). Deadline for submissions: April 7, 2008 (abstracts), April 14, 2008 (full papers).
♦ Oct 26-30	2008 ACM **SIGAda Annual International Conference** (SIGAda'2008), Portland, Oregon, USA. Sponsored by ACM SIGAda. Topics include: Transitioning to Ada 2005; Educational challenges for developing reliable, safe, secure software; Ada and SPARK in the classroom and student laboratory; Language selection for a high reliability system; Use of high reliability subsets or profiles such as MISRA C, Ravenscar, SPARK; High reliability standards and their issues; Software process and quality metrics; Analysis, testing, and validation; Use of ASIS for new Ada tool development; Mixed-language development; High-reliability development experience reports; Static analysis of code; Integrating COTS software components; System Architecture & Design; Information Assurance; Ada products certified against Common Criteria / Common Evaluation Methodology; etc. Deadline for submissions: May 12, 2008 (technical articles, extended abstracts, experience reports, workshops, panel sessions, and tutorials).
☺ November 10-12	10th **International Symposium on Distributed Objects, Middleware and Applications** (DOA'2008), Monterrey, Mexico. Topics include: Application case studies of distribution technologies; Development methodologies for distributed applications; Interoperability with other technologies; Reliability, fault tolerance, quality-of-service, and real time support; Scalability and adaptivity of distributed architectures; Software engineering for distributed middleware systems; etc. Deadline for submissions: June 8, 2008 (abstracts), June 15, 2008 (papers).
☺ December 01-04	9th **International Conference on Parallel and Distributed Computing, Applications, and Techniques** (PDCAT'2008), Dunedin, New Zealand. Topics include: Parallel/distributed architectures; Multi-core related technologies; Reliability, and fault-tolerance; Formal methods and programming languages; Software tools and environments; Parallelizing compilers; Component-based and OO Technology; Parallel/distributed algorithms; Task mapping and job scheduling; Security and privacy; etc. Deadline for submissions: April 15, 2008 (workshops), June 1, 2008 (papers), June 30, 2008 (tutorials).
December 01-05	ACM/IFIP/USENIX 9th **International Middleware Conference** (Middleware'2008), Leuven, Belgium. Topics include: design, implementation, deployment, and evaluation of distributed system platforms and architectures for future computing and communication environments. Deadline for submissions: April 23, 2008 (abstracts), April 30, 2008 (papers).
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!
☺ December 10-12	6th **International Symposium on Parallel and Distributed Processing with Applications** (ISPA'2008), Sydney, Australia. Topics include: all aspects of Parallel/Distributed Computing and Networking, and their applications, such as Parallel/distributed system architectures, Tools and environments for software development, Distributed systems and applications, Reliability, fault-tolerance, and security, etc. Deadline for submissions: June 1, 2008 (papers)

Preliminary Call for Participation

13th International Conference on Reliable Software Technologies – Ada-Europe 2008

16-20 June 2008, Venice, Italy

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

The Conference

The 13th International Conference on Reliable Software Technologies – Ada-Europe 2008 will take place in Venice, Italy, on 16-20 June 2008. Following its consolidated tradition, the conference will span a full week, with at its centre from Tuesday to Thursday a three-day technical program accompanied by vendor exhibitions, and at either end on Monday and Friday a string of parallel tutorials. The previous twelve editions in this conference series were 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) and Geneva, Switzerland ('07).

The conference has 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 analysis, design, programming, verification and maintenance of long-lived, high-integrity software systems for a variety of application domains. The program also features outstanding keynotes and a robust track of industrial presentations. As usual, the conference provides ample time for Q&A sessions, panel discussions and social events. Participants include practitioners and researchers from industry, academia and government organizations interested in the promotion and development of reliable software technologies.

13th INTERNATIONAL CONFERENCE ON RELIABLE SOFTWARE TECHNOLOGIES
16-20 JUNE 2008

Overall Program

The conference program altogether features 10 tutorials, a technical program of 20 thoroughly refereed papers, a collection of 12 industrial presentations reflecting current practice and challenges in real-life software projects, a special track on software engineering education, three eminent invited speakers, a rich exhibition, and an especially attractive social program. Springer will publish the proceedings of the regular program of conference as Volume 5026 of the LNCS. The Ada User Journal will publish the proceedings of the other tracks of the program.

Keynote Addresses

Three eminent keynote speakers have been selected to open each day of the core conference program:

- Alberto Sangiovanni-Vincentelli (University of California at Berkeley, USA), a most authoritative member of the embedded systems community, will deliver a talk entitled: *Embedded Software Design: Art or Science?*
- Robert Dewar (New York University, USA), a worldwide expert in programming technologies, will discuss where programming languages are expected to go next in a talk evocatively entitled: *Lost in Translation.*
- Christian Queinnec (Lip6 Paris, France), a leading researcher in reliable software technologies, will explore the inner heart of the Service-Oriented Architecture in a talk entitled: *Three ways to improve SOA reliability.*

Tutorial Program

The two days at either side of the core conference program include a rich selection of tutorials delivered by domain experts who will cover a variety of topics of interest to the conference community:

- *AADL: Architecture Analysis and Design Language*, Jean-Pierre Rosen (Adalog, France)
- *The best of Ada 2005*, John Barnes (John Barnes Informatics, UK)
- *Object-Oriented Programming in Ada 2005*, Matthew Heaney (On2 Technologies, USA)
- *Preserving Model-Asserted Real-Time Properties at Execution Level for High-Integrity Systems*, Tullio Vardanega (University of Padua, Italy), Juan Antonio de la Puente (Technical University of Madrid, Spain)
- *Technical Basis of Model Driven Engineering* and
Verification Techniques for Dependable Systems, both by William Bail (The MITRE Corporation, USA)
- *A Practical Introduction to Model-Driven Software Development using Eclipse*, Cristina Vicente-Chicote and Diego Alonso-Cáceres (Universidad Politécnica de Cartagena, Spain)
- *Languages for Safety-Critical Software: Issues and Assessment*, Benjamin Brosgol (AdaCore, USA)
- *Service-Oriented Architecture Concepts and Implementations*, Ricky Sward (The MITRE Corporation, USA)
- *About Real-Time Scheduling Analysis of Ada Applications*, Frank Singhoff (University of Brest, France).

Technical Program

The technical program of the conference includes 20 thoroughly peer-refereed papers on a wealth of subjects pertinent to the conference themes, from submissions coming from over 20 countries worldwide. The program also features a day-long collection of 12 industrial presentations of challenges faced and solutions devised in real-life projects, a half-day track focusing on software engineering education and a half-day vendor-presentation session.

Exhibition

The exhibition will open in the mid-morning break on Tuesday and run continuously until the end of the afternoon break on Thursday. Coffee breaks and exhibitions will both be held in a spectacular cloister area in the inner heart of the conference centre. Breaks will last one full hour to allow attendees comfortable time to visit the exhibition.

Conference Venue

Venice is a marvel that words can't explain. You really want to wander its alleys and enjoy the experience of its floating on the water. We are truly fortunate at being able to host the first coming of the conference in Italy, at a 10-minute walk from piazza San Marco, the most renowned centre of Venetian urban architecture, which still bears the signs of amazingly thriving cultural and commercial life when the Venetian republic was the gateway of Europe to the Far East. The conference centre, built on a fully-refurbished and modernly equipped restoration of a XVI-century convent, is located at the "Zattere" overlooking the Giudecca island, on the South-South-East angle of Venice, just behind the Accademia, which holds treasures of art.

Venice has the third most visited airport in Italy, directly connected to all capital cities in Europe, and to some major cities in the USA and Asia. The airport location offers a majestic view of the lagoon on many landing and take-off routes. Venice is connected to the airport via both water and land transport.

Social Program

The social program of the conference will open with a welcome reception at Palazzo Loredan-Franchetti, a three-storied patrician villa overlooking the Grand Canal, which hosts the historic premise of the regional institute for science, literature and art. The reception will be accompanied by musical entertainment offered by distinguished members of the conference community.

The conference banquet will take place in the island of Torcello, the farthest island of the lagoon, just past picturesque Burano. The Torcello island which used to be vastly populated at the time of the Venetian republic has lost almost all of its population but kept its beauty and natural, cultural and historical attraction. The conference participants will ride on a private boat along the Grand Canal to the renowned "Osteria del Diavolo" restaurant at Torcello. The journey will be accompanied by appetizers and musical entertainment, as well as by the spectacular scenery of the lagoon itself viewed first at sunset and then in the fullness of night on the return leg.

In cooperation with SIGAda acm Association for Computing Machinery

For the latest information on the conference consult: `http://www.ada-europe.org/conference2008.html`

Association for Computing Machinery

Call for Technical Contributions

Submission Due Date: May 12, 2008

SIGAda Annual International Conference: Toward Safe, Secure, Reliable Software

October 26-30, 2008

University Place Hotel and Conference Center, Portland, Oregon, USA

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

(ACM Approval Pending)

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.

CONFERENCE LOCATION: Portland is the attractive, livable "City of Roses" in the Pacific Northwest. The weather in October is usually cool and often beautiful. University Place is a modern and reasonably-priced hotel located within walking distance of the central business district, the lively riverfront area, and the Portland State University campus.

HOW YOU CAN CONTRIBUTE: SIGAda 2008 solicits contributions in six major categories: Technical Articles, Extended Abstracts, Experience Reports, Workshops, Panel Sessions, and Tutorials. Contributions from students and faculty are actively solicited. Final acceptance will be contingent on a commitment to present the contribution at the Conference.

POSSIBLE TOPICS include but are not limited to:

- Transitioning to Ada 2005
- Educational 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
- Analysis, testing, and validation
- Use of ASIS for new Ada tool development
- High-reliability development experience reports
- Static analysis of code
- Integrating COTS software components
- System Architecture & Design
- Information Assurance
- Ada products certified against Common Criteria / Common Evaluation Methodology

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

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 **May 12, 2008**, in Word, PDF, or text format as follows:

Technical Articles, Extended Abstracts, Experience Reports, and Panel Session Proposals: Program Chair, Leemon C. Baird III (leemon.baird@usafa.edu).

Workshop proposals: Workshops Chair, Bill Thomas (Bthomas@MITRE.org).

Tutorial proposals: Tutorials Chair, David A. Cook (Dcook@AEgisTG.Com).

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 S. Ron Oliver (SROliver@CSC.CalPoly.Edu) or Greg Gicca (gicca@adacore.com) for information about becoming a sponsor and/or exhibitor at SIGAda 2008.

IMPORTANT INFORMATION FOR NON-US SUBMITTERS:

General Visa Information: The sites http://www.UnitedStatesVisas.gov and http://travel.state.gov have information about obtaining a visa for those traveling to the United States. Both sites have links to websites for U.S. embassies and consulates worldwide. The embassy and consulate websites have helpful information about procedures, timelines, communities served, required documentation, and fees.

Letters from ACM: International registrants should be particularly aware and careful about visa requirements, and should plan travel well in advance. All visa inquiries must be handled by ACM Headquarters. Please send your request for a letter in support of a visa application to Ashley Cozzi (acozzi@acm.org), and include your name, mailing address, and fax number, as well as the name of the conference you are attending. Authors should also include the title of their contribution. Please note that ACM does not issue formal "letters of invitation" to any of its conferences.

Please submit any questions on the conference to the Conference Chair, Michael Feldman (mfeldman@gwu.edu).

CONFERENCE COMMITTEE:

Conference Chair
Michael Feldman
George Washington Univ. (retired)
Publicity
Ron Price
Consultant
Workshops
Bill Thomas
The MITRE Corporation
Registration
Thomas A. Panfil
US Department of Defense
SIGAda Chair
John W. McCormick
University of Northern Iowa

Program Chair
Leemon C. Baird III
US Air Force Academy
Treasurer
Martin C. Carlisle
US Air Force Academy
Proceedings and Webmaster
Clyde Roby
Institute for Defense Analyses
Tutorials
David A. Cook
AEgis Technologies Group, Inc
SIGAda Vice Chair, Mtgs & Confs
Ricky E. Sward
The MITRE Corporation

Exhibits and Sponsors
S. Ron Oliver
caress Corporation
Exhibits and Sponsors
Greg Gicca
AdaCore
Local Arrangements
Elizabeth Adams
James Madison University
Local Arrangements
Geoff Smith
Lightfleet Corporation

Practical Application of Static Analysis for Embedded Systems

David N. Kleidermacher

Green Hills Software, Inc.; email: davek@ghs.com.

Abstract

Static analysis is a promising technique for improving reliability and security of software and systems. Static analysis tools analyze software to find flaws that may go undetected using traditional techniques, such as compilers, human code reviews, and testing. A number of limitations, however, have prevented widespread adoption in everyday embedded software development. Static analysis tools often take too long to execute, are not well integrated into the software development environment, and are expensive to procure. In addition, static analysis tools have not been effectively applied to problems, such as multi-thread synchronization and stack overflow errors, that are specific to embedded systems. Furthermore, static analysis techniques should be extended to include enforcement of coding standards that are commonly used in high reliability embedded software development processes. This paper will introduce a new approach to static analysis that solves many of these problems. Specific metrics will be provided to demonstrate how the new approach makes the use of static analysis tools practical and effective for everyday embedded software development.

1 Introduction

Static source code analyzers attempt to find code sequences that when executed could result in buffer overflows, resource leaks, or many other security and reliability problems. Source code analyzers are effective at locating a significant class of flaws that are not detected by compilers during standard builds and often go undetected during run-time testing as well.

Earlier is Better

A number of studies over the years have shown that the cost of detecting and correcting a software flaw increases dramatically as a project moves through the development, integration, quality assurance, and deployment cycle [1], as depicted in figure 1. This reality matches common sense: a software developer who finds his own bug soon after adding it has recent context in which to quickly understand and fix the problem. As a project enters integration and test phases, a flaw is often discovered by someone other than the developer who added it and often much later than the flaw was introduced. This of course makes it more difficult to trace the flaw back to its source and for developers to infer the cause and determine the optimal solution to the problem. Once a product has been deployed, the cost of a serious flaw is bloated by customer service resource usage, patching protocols, recalls, and returns.

From the cost perspective, static analysis is one of the most powerful tools in the software developer's arsenal because it enables flaws to be cheaply discovered and fixed well before even a single line of code is ever executed.

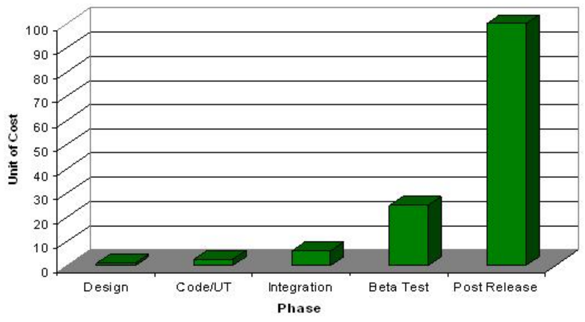


Figure 1 Cost of software flaws

Software Complexity

Many of the problems relating to loss in quality and safety in software can be attributed to the growth of complexity that cannot be effectively managed [2]. For instance, desktop operating system code bases have been increasing at a staggering rate. Microsoft Windows went from 6M lines of code in 1993, to 29M in 2000, to 50M in 2005. Debian Linux increased even more rapidly: over 55M loc in 2000, to 104M in 2002, to 215M in 2005 [3].

Embedded systems are suffering widely from this complexity boom, as faster microprocessors and customer demand provide a fertile ground for software-based multimedia, connectivity, and overall intelligence enhancement. Security flaws are yet another example of the loss in quality resulting from complexity; according to CERT statistics, the number of documented vulnerabilities has been increasing almost exponentially, from approximately 400 in 1999, to more than 4000 in 2002, and more than 8000 in 2006 [4].

Complexity strains traditional reliability techniques such as code reviews and implies a growing necessity for automated static analysis tools.

Common Features of Static Analysis Tools

Most source code analyzers perform a full program analysis, finding bugs caused by complex interactions between pieces of code that may not even be in the same source file.

The analyzer determines potential execution paths through code, including paths into and across subroutine calls, and how the values of program objects (such as standalone variables or fields within aggregates) could change across these paths. The objects could reside in memory or in machine registers.

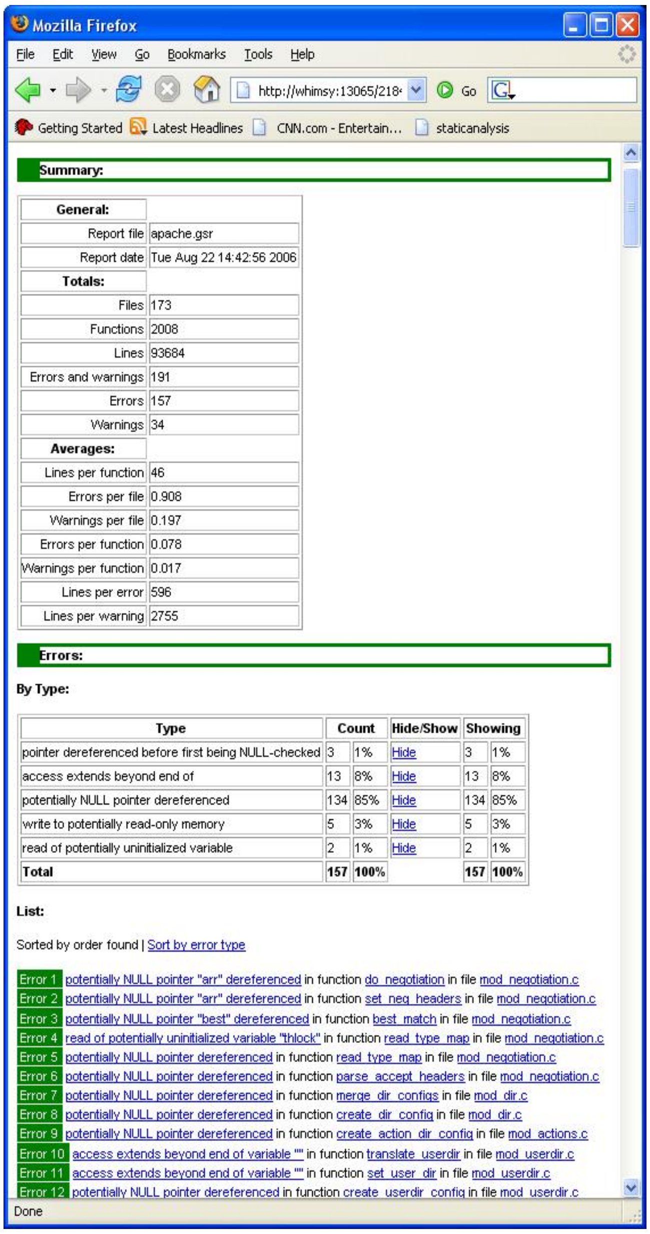
The analyzer looks for many types of flaws. It looks for bugs that would normally compile without error or warning. The following is a list of some of the more common errors that an analyzer will detect:

- Potential NULL pointer dereferences
- Access beyond an allocated area (e.g. array or dynamically allocated buffer); otherwise known as a buffer overflow
- Writes to potentially read-only memory
- Reads of potentially uninitialized objects
- Resource leaks (e.g. memory leaks and file descriptor leaks)
- Use of memory that has already been deallocated
- Out of scope memory usage (e.g. returning the address of an automatic variable from a subroutine)
- Failure to set a return value from a subroutine
- Buffer and array underflows

The analyzer often has knowledge about how many standard runtime library functions behave. For example it knows that subroutines like free should be passed pointers to memory allocated by subroutines like malloc. The analyzer uses this information to detect errors in code that calls or uses the result of a call to these functions.

The analyzer can also be taught about properties of user-defined subroutines. For example if a custom memory allocation system is used, the analyzer can be taught to look for misuses of this system. By teaching the analyzer about properties of subroutines, users can reduce the number of false positives. A false positive is a potential flaw identified by the analyzer that could not actually occur during program execution. Of course, one of the major design goals of static analyzers is to limit the number of false positives so that developers can minimize time looking at them. If an analyzer generates too many false positives, it will become irrelevant because the output will be ignored by engineers. However, since an analyzer is not able to understand complete program semantics, it is not possible to totally eliminate false positives. In some cases, a flaw found by the analyzer may not result in a fatal program fault, but could point to a questionable construct that should be fixed to improve code clarity. A good example of this is a write to a variable that is never subsequently read.

A common output format of a static analysis tool is a set of web pages hosted by a web server. Green Hills Software's DoubleCheck™ analyzer is also capable of emitting errors



Summary:

General:	
Report file	apache.gsr
Report date	Tue Aug 22 14:42:56 2006
Totals:	
Files	173
Functions	2008
Lines	93684
Errors and warnings	191
Errors	157
Warnings	34
Averages:	
Lines per function	46
Errors per file	0.908
Warnings per file	0.197
Errors per function	0.078
Warnings per function	0.017
Lines per error	596
Lines per warning	2755

Errors:

By Type:

Type	Count		Hide/Show	Showing	
pointer dereferenced before first being NULL-checked	3	1%	Hide	3	1%
access extends beyond end of	13	8%	Hide	13	8%
potentially NULL pointer dereferenced	134	85%	Hide	134	85%
write to potentially read-only memory	5	3%	Hide	5	3%
read of potentially uninitialized variable	2	1%	Hide	2	1%
Total	**157**	**100%**		**157**	**100%**

List:

Sorted by order found | Sort by error type

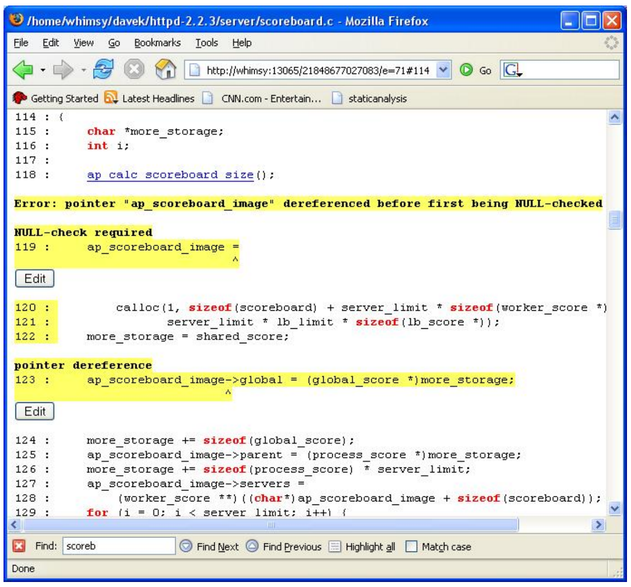
Error 1 potentially NULL pointer "arr" dereferenced in function do_negotiation in file mod_negotiation.c
Error 2 potentially NULL pointer "arr" dereferenced in function set_neg_headers in file mod_negotiation.c
Error 3 potentially NULL pointer "best" dereferenced in function best_match in file mod_negotiation.c
Error 4 read of potentially uninitialized variable "thlock" in function read_type_map in file mod_negotiation.c
Error 5 potentially NULL pointer dereferenced in function read_type_map in file mod_negotiation.c
Error 6 potentially NULL pointer dereferenced in function parse_accept_headers in file mod_negotiation.c
Error 7 potentially NULL pointer dereferenced in function merge_dir_configs in file mod_dir.c
Error 8 potentially NULL pointer dereferenced in function create_dir_config in file mod_dir.c
Error 9 potentially NULL pointer dereferenced in function create_action_dir_config in file mod_actions.c
Error 10 access extends beyond end of variable "" in function translate_userdir in file mod_userdir.c
Error 11 access extends beyond end of variable "" in function set_user_dir in file mod_userdir.c
Error 12 potentially NULL pointer dereferenced in function create_userdir_config in file mod_userdir.c

Figure 2 Web page flaw summary

as part of the build process. The web interface enables the user to browse high level summaries of the different flaws found by the analyzer (Figure 2) and then click on hyperlinks to investigate specific problems. Within a specific problem display, the error is usually displayed inline with the surrounding code, making it easy to understand (Figure 3). Function names and other objects are hyperlinked for convenient browsing of the source code. Since the web pages are running under a web server, the results can easily be shared and browsed by any member of the development team.

Open Source Static Analysis

As a later example will show, current generation open source static analysis tools such as lint and splint suffer from high false positive rates, making them impractical for use on complex software projects. The analysis time required to sift through the false positives often far



```
{
    char *more_storage;
    int i;

    ap calc scoreboard size();

Error: pointer "ap_scoreboard_image" dereferenced before first being NULL-checked

NULL-check required
    ap_scoreboard_image =
                        ^
Edit

        calloc(1, sizeof(scoreboard) + server_limit * sizeof(worker_score *)
                server_limit * lb_limit * sizeof(lb_score *));
    more_storage = shared_score;

pointer dereference
    ap_scoreboard_image->global = (global_score *)more_storage;
                        ^
Edit

    more_storage += sizeof(global_score);
    ap_scoreboard_image->parent = (process_score *)more_storage;
    more_storage += sizeof(process_score) * server_limit;
    ap_scoreboard_image->servers =
        (worker_score **)((char*)ap_scoreboard_image + sizeof(scoreboard));
    for (i = 0; i < server_limit; i++) {
```

/home/whimsy/davek/httpd-2.2.3/server/scoreboard.c - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://whimsy:13065/21848677027083/e=71#114
Go
Getting Started
Latest Headlines
CNN.com - Entertain...
staticanalysis
Find: scoreb
Find Next
Find Previous
Highlight all
Match case
Done

Figure 3 Context-sensitive flaw display

outweighs the cost of more accurate commercial static analysis tools.

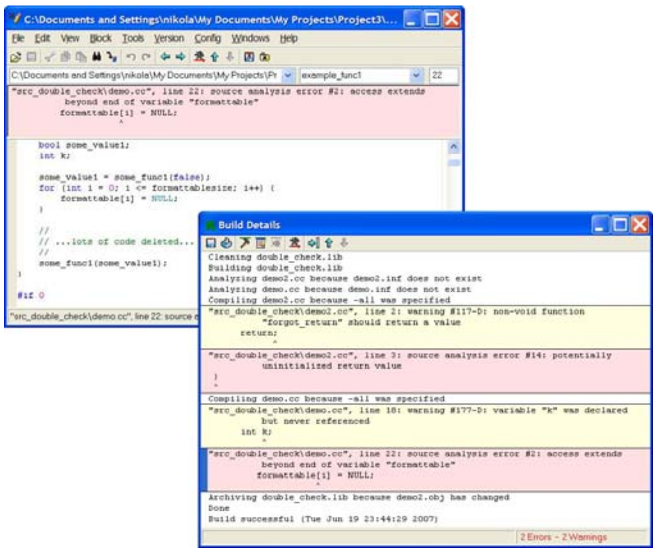
2 Barriers to Widespread Adoption

A recent survey found that less than 5% of embedded systems engineers make regular use of static analysis tools [5]. Engineers cited several barriers to adoption: poor integrated development environment (IDE) integration; prohibitive execution time; lack of features specific to embedded systems; and cost.

IDE Integration

Commercial static analyzers often run as separate tools, distinct from the tool chain used to develop and build application software. Thus, users must separately install, license, and configure the analyzer. Configuration can often be time consuming, requiring days of customization in order to cajole the analyzer into processing the user's particular dialect of source code.

Green Hills Software's DoubleCheck™ analyzer introduces a new approach in which the static analysis is performed within the same compiler used to build software. This approach brings with it several advantages, including the obvious benefit of reducing the time to effective usage. Since static analysis is performed by the compiler, it doesn't need to make guesses regarding the proper application of build options, location of include header files and directories, or the definitions of preprocessor macros. Flaws discovered by the static analyzer can be interleaved with the other standard diagnostics output by the compiler (figure 4). Furthermore, common IDE integrations between the project builder and the editor augment the usability of the static analysis tool: when a static analysis flaw is reported during the build process, the user can hyperlink from the builder's output window back to the source code quickly, rectify the error, and then return to building the program.



```
"src_double_check\demo.cc", line 22: source analysis error #2: access extends
        beyond end of variable "formattable"
      formattable[i] = NULL;
                 ^

    bool some_value1;
    int k;

    some_value1 = some_func1(false);
    for (int i = 0; i <= formattablesize; i++) {
        formattable[i] = NULL;
    }

    //
    // ...lots of code deleted...
    //
    some_func1(some_value1);
}

#if 0
```

```
Cleaning double_check.lib
Building double_check.lib
Analyzing demo2.cc because demo2.inf does not exist
Analyzing demo.cc because demo.inf does not exist
Compiling demo2.cc because -all was specified
"src_double_check\demo2.cc", line 2: warning #117-D: non-void function
        "forgot_return" should return a value
    return;
          ^
"src_double_check\demo2.cc", line 3: source analysis error #14: potentially
        uninitialized return value
 }
 ^
Compiling demo.cc because -all was specified
"src_double_check\demo.cc", line 18: warning #177-D: variable "k" was declared
        but never referenced
    int k;
        ^
"src_double_check\demo.cc", line 22: source analysis error #2: access extends
        beyond end of variable "formattable"
      formattable[i] = NULL;
                 ^
Archiving double_check.lib because demo2.obj has changed
Done
Build successful (Tue Jun 19 23:44:29 2007)
```

C:\Documents and Settings\nikola\My Documents\My Projects\Project3\...
File Edit View Block Tools Version Config Windows Help
C:\Documents and Settings\nikola\My Documents\My Projects\Pr
example_func1
22
"src_double_check\demo.cc", line 22: source
Build Details
2 Errors - 2 Warnings

Figure 4 Integration of static analysis with project builder

With proper builder integration, static analysis becomes merely another option that any developer can easily enable or disable from the options menu.

Execution Time

Many commercial static analyzers require orders of magnitude more execution time than a regular compile. Large software projects may require hours or even days of analysis time. This execution cost presents a barrier to adoption. Static analysis, if used at all, will only be employed periodically or during test phases. Knowing that software development cost and time to market decrease when flaws are detected earlier in the project cycle, it follows that static analysis tools could enjoy improved adoption if execution time can be reduced to a level that encourages constant use; developers can detect flaws while software is first written and before it is ever committed to a configuration management system.

Here again is where the integrated compiler approach proves beneficial. DoubleCheck's analysis engine takes advantage of dataflow analysis, constant propagation, and path pruning algorithms, developed and tuned for execution time efficiency over many years to perform complex code optimizations. The result is that the integrated analyzer performs much better than traditional standalone analyzers.

Secondly, the total time to build and analyze software is reduced since the compiler uses a single parsing pass of the code to perform both compilation and analysis.

Thirdly, the integration with the IDE enables the analyzer to take advantage of the IDE's existing distributed build mechanism. The parsing pass for the project's source code is distributed across idle workstation assets on a user's network, dramatically reducing the total analysis time.

Finally, the compiler and project builder's dependency analysis system monitors which sources have changed since the last build to analyze only those functions that matter, thereby reducing analysis time throughout the project development cycle.

Studies have shown that the integrated compiler approach to static analysis yields analysis times comparable with traditional compilation times, thereby removing an important barrier to adoption.

Lack of Focus for Embedded Systems

Many static analyzers were designed for enterprise software development projects and therefore ignore some of the thorny problems, such as run-time stack overflow and multi-thread synchronization problems that plague embedded software developers.

Stack overflow detection presents another problem addressed by the integrated compiler approach: when the tool generates code, the stack sizes are known for each subroutine, enabling the analyzer to compute worst case stack size requirements and warn the user when these requirements exceed the programmer's per-thread allocations.

Since embedded systems often employ real-time operating systems (RTOS), the static analysis tools should incorporate RTOS-awareness. For example, the static analysis tool should understand the RTOS application programming interface (API) so that thread creation functions and their parameters (e.g. entry point, stack size) are parsed and used to automatically detect stack overflows without requiring any special user customization. RTOS-specific resources, such as mutexes, message queues, and I/O devices, can be automatically tracked to avoid improper deallocations and leaks. RTOS awareness enables a static analyzer to address important, and often subtle, flaws that plague embedded system reliability.

3 Case Study

According to apache.org, the *Apache* open source hypertext transfer protocol (HTTP) server is the most popular web server in the world, powering more than 70% of the web sites on the Internet [6]. Given the ubiquity of *Apache* and the world's dependence on the Internet, the reliability and security of Apache represent an important concern for all of us. A serious flaw in Apache could cause widespread inconvenience, financial loss, or worse.

The *Apache* web server consists of approximately 150,000 lines of code. This code has been analyzed by a number of static analysis tools. Coverity recently published results of their analysis, showing 22 errors and 10 minutes of analysis time [7].

Apache Flaws Found

Green Hills Software's DoubleCheck analyzer reported a larger number of flaws: 140. DoubleCheck found a significant number (126) of potential NULL dereferences such as the following from line 120 in Apache source file scoreboard.c:

```
ap_scoreboard_image = calloc(1,  sizeof(scoreboard) +
                        server_limit * sizeof(worker_score *) +
                        server_limit * lb_limit * sizeof(lb_score *));
```

Clearly, this allocation of memory could be substantial. It would be a good idea to make sure that the allocation succeeds before referencing the contents of ap_scoreboard_image. However, soon after the allocation statement, we have this use:

```
ap_score_board_image->global =
                (global_score *) more_storage;
```

The dereference is unguarded, making the application susceptible to a fatal crash. Another example can be found at line 765 in the file mod_auth_digest.c:

```
entry = client_list->table[idx];
prev  = NULL;
while (entry->next){  /* find last entry */
  prev  = entry;
  entry = entry->next;
  …
 }
```

Note that the variable entry is unconditionally dereferenced at the beginning of the loop. This alone would not cause the analyzer to report an error. At this point in the execution path, the analyzer has no specific evidence or hint that entry could be NULL or otherwise invalid. However, the following statement occurs after the loop:

```
if (entry) {
  …
}
```

By checking for a NULL entry pointer, the programmer has indicated that entry could be NULL. Tracing backwards, the analyzer now sees that the previous dereference to entry at the top of the loop is a possible NULL reference.

Apache Analysis Time

Executing on the same hardware platform documented in the Coverity report, DoubleCheck required less than two minutes of execution time, approximately a factor of five faster. When the distributed analysis feature was enabled, analysis time was further reduced to approximately 30 seconds.

As part of this research project, Green Hills Software engineers also ran the standard Linux-hosted *splint* analysis tool on the Apache web server source base, resulting in a report of 19197 errors, a false positive rate beyond practical use.

4 Conclusion

Most high reliability development processes espouse the use of a coding standard that governs how developers write code [8]. The goal of the coding standard is to increase reliability by promulgating intelligent coding practices. For example, a coding standard may contain rules that help developers avoid dangerous language constructs, limit complexity of functions, and use a consistent syntactical and commenting style. These rules can drastically reduce the occurrence of flaws, make software easier to test, and improve long term maintainability.

Static analysis represents the next major ingredient to be added to high quality coding standards. The integrated compiler approach makes it easy and efficient to incorporate automated static source code checking into the everyday software development process.

References

[1] The Economic Impacts of Inadequate Infrastructure for Software Testing, NIST, May 2002, p. 1-13.

[2] Parker, R., *Are Vendors Doing Enough To Improve Software?*, Optimize Magazine. Available at: http://www.optimizemag.com/issue/009/squareoff.htm

[3] *Wikipedia - Source lines of code*. Available at: http://en.wikipedia.org/wiki/Source_lines_of_code

[4] *CERT Statistics*. Available at: http://www.cert.org/stats

[5] *Informal survey of 100 engineers at Embedded World 2006*, Nuremberg, Germany.

[6] http://httpd.apache.org

[7] Chelf, B., *Measuring Software Quality: A Study of Open Source Software*, February 2007.

[8] *DO-178B, Software Considerations in Airborne Systems and Equipment Certification*, RTCA inc, 1992.

Contact

David Kleidermacher is Chief Technology Officer at Green Hills Software where he has been designing compilers, software development environments, and operating systems for the past sixteen years. David is responsible for the company's product planning, development, deployment, and technical support. David holds a bachelor of science in computer science from Cornell University. Contact David at davek@ghs.com.

13th International Real-Time Ada Workshop

17-19 April 2007
Woodstock, Vermont
USA

Session: Programming Patterns and Libraries

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

Program Committee

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

[a] Program Chair [b] Local Chair

Workshop Participants

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

Sponsors

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

Session: Programming Patterns and Libraries

Chair: Michael González Harbour

Rapporteur: J. Javier Gutiérrez

1 Session Goals

The chairman introduced the session pointing out its main goals, which were to:

- discuss different visions of architectural frameworks and coding patterns with Ada 2005
- and formulate proposals and roadmaps to facilitate adoption or secondary standards.

Afterwards, the three papers [1][2][3] included in this session were briefly introduced by their authors followed by the discussion on the issues proposed in them.

2 Real-Time Utilities

Andy Wellings introduced the topic included in paper [1] by showing how real-time utilities can be provided as extensions to Ada, looking at previous experiences coming for example from Real-Time Java, which has a lot of available utilities. It was stated that, from the beginning, Ada 83 provided good and relatively simple low-level mechanisms for programming those kinds of systems, but considering the complexity of the new mechanisms introduced in the Ada 2005 and the requirements of modern applications it is desirable to provide higher-level utilities. Modern programming languages follow this path so there is a need for having this support also in Ada, perhaps through a secondary standard or some other dissemination process.

Once the initial position was set, Andy talked about the following issues that an application may require: templates (for periodic, sporadic or aperiodic tasks), to limit the amount of CPU-time that a task can receive by its association to an execution-time server, what happens in the case of a deadline miss or an execution time overrun, and the nature of the mode changes.

The proposed approach splits the support for the task into four components, each one encapsulated in a package: the application functionality (Real_Time_Task_State package), the mechanisms needed to control the release of real-time tasks and to detect the deadline missed or execution time overruns (Release_Mechanisms package), the response of tasks to deadline misses or execution time overruns (Real_Time_Tasks package), and the mechanisms to ensure that a subsystem is only given a certain amount of the CPU resource (Execution_Server package).

The specification of the Real_Time_Task_State package defines the Task_State type as the root of a class with operations to execute on each release of the task and when a deadline miss or an execution time overrun occurs. Then the inherited classes for periodic, sporadic and aperiodic tasks can be derived (see the position paper [1] for details). The main idea is that this structure is provided to the application developer as a ready-to-use infrastructure so that he or she can just concentrate on the functionality. An initial comment was made to propose the usage of interfaces to implement the state of a task. The difficulty of having public data attributes in the specification could be solved by adding set/get operations to manage these attributes.

The implementation of the release mechanisms specifies every release mechanism as a synchronized interface derived from the root type and separated in child packages for only deadline miss notification, only overrun detection, or both. The execution time servers were briefly introduced as the mechanisms capable of guaranteeing the usage of a fixed amount of the CPU resource for periodic, aperiodic or sporadic tasks. Finally, the action tasks for this framework are implemented via task templates; an example of a real-time task with deadline termination was shown. This finished the presentation of the position paper and led to an initial discussion.

The following topics were pointed out as part of this discussion:

- A question about whether it is possible or not to pass some data on the release.
- The possibility of addressing mode changes. Information on the current mode might be included in the release code, or could be added as an additional parameter to the operation Inform_Of_a_Deadline_Miss, although this could allow extensions in ways other than expected and become more complex.
- The possibility of having several kinds of notification. It was stated that with the individual handling of notifications and their possible combinations, adding more kinds in the future would make the framework become much more complex.
- A question on whether a pattern such as x over y times of deadline misses can be constructed into this framework.
- The possibility of having a matrix of pending notifications or some hierarchy on the notification treatment.

At this point it was suggested to go to other presentations and come back later to a more general discussion.

3 Programming Servers in Ada 2005

Alan Burns presented the second position paper [2], which deals with the issue of programming execution-time servers

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

in Ada 2005. He introduced the new features of Ada 2005 to control the execution time of a group of tasks managed together and to stop all of them if some budget is exhausted, and showed an example for a deferrable server running all client tasks (previously registered) at the same fixed priority.

Afterwards, he set the aim of their proposal to classify different types of servers that programmers might wish to use, in order to produce a library of these servers. The types of server should be classified according to the following characteristics:

- Dispatching: all clients have the same priority (serial), servers have a unique range of priorities (concurrent), or each client has a priority (similar to the Java model with processing groups, which is difficult to analyze). A comment was made to take into account the possibility of having only one client per server.
- Scheduling: Fixed Priorities, EDF, and perhaps mixed schemes.
- Replenishment: periodic, or related to usage (limiting or not the level of concurrency).
- Identifying sessions: to know if a client is active or not in the server, or to notify the server when a client is active, when it blocks, and when it ends its execution for the current instance. It should be useful for soft real-time.
- Whether a client should be informed or not when the budget is exhausted.
- Client binding: whether the client should follow the behaviour of the server or not.
- Different models for capacity sharing.

As a conclusion, the motivation is to see if Ada 2005 is useful for programming these issues, and to identify different server types in addition to deferrable, sporadic, periodic, constant bandwidth preserving, etc.

Once the presentation was finished, it was asked if the servers could be integrated into the previously discussed framework. The answer was that it could be addressed once the mechanism for session control is defined and the details of its integration with the framework are solved. It was suggested that priority bands could emulate a server by attaching the task to a band. Finally, it was expressed that the main idea is to put together different types of dispatching and scheduling policies, and that the same reasons that move people to use Ada, could move them to use the frameworks.

4 Ada 2005 Code Patterns for MDA

The third position paper [3] was presented by Tullio Vardanega, who talked about a 4-view MDA (Model Driven Architecture) design space. The notion is quite similar to that presented in [1], i.e., to propose templates and a framework that programmers can follow. This framework has been designed to support the usage of Ada in applications oriented to the Ravenscar profile. The main goal is to allow the designer of an application to concentrate on its functional parts, i.e., the designer only needs to worry about the sequential parts of the code, and not about other aspects like concurrency for example.

The framework gives an interface of a model to build the application. This is made via the Application-Level Containers which hold the functional modeling of the application. These containers are not executable but they are all that the designer needs to work with. The framework has other types of containers, called Virtual Machine-Level Containers, into which the Application-Level ones are mapped, and that support the execution aspects of the framework. The idea is to show if the whole process is correct, i.e., if the original structure corresponds to the final application.

Tullio showed an example about how concurrency can be overlaid. In this example the designer only has to write an operation corresponding to the functional view of the framework, while the rest of the views are offered as templates where nothing has to be coded. This way, the operation is a method inserted into a complex framework (very similar to a components framework). Again, the main idea is not to involve the user in the structural Ravenscar or real-time part. The proposed framework is currently working on an UML and Eclipse environment, fully automated and producing Ravenscar code. Another idea is to explore the use of Ada interfaces instead of generics.

5 Discussion

Following the introduction of the three position papers made by their authors, a main discussion started with an initial question by the chair of the session related to how to proceed, i.e., if some kind of standard should be produced. The idea is to produce a unique framework probably proposed as a secondary standard. The comments that this question arisen can be summarized as follows:

- The question is whether or not we can do it and how it can be funded.
- A comment to focus on paradigms and not only on the language.
- To find funding for the different developments, it would be a good idea to have an initial comparison, looking at similar efforts such as those by the companies exploring Java. It was suggested that this would be a second step and before having a full implementation, people could contribute by implementing parts of the framework (as student projects or things like that).
- It was expressed that it was early to propose the frameworks as a secondary standard.
- It was asked if anybody was familiar with the Container Library, and the process by which it had been standardized. We could follow the same path as a second step, and perhaps ARTIST could give support to the first step.
- It was suggested to use the framework proposed in [1], perhaps borrowing ideas from the framework proposed in [3], and trying to integrate the servers presented in [2].

- It was suggested there was a need for minimal documentation in order for other people to contribute to the project with new servers or components, and to be familiar with the framework.

The discussion then focused on Andy's slides to dwelve deeper into the use of an interface for the definition of the Task_State in the Real_Time_Task_State package instead of an object with attributes and methods. The issue brought up a lot of comments and questions that can be summarized as follows:

- A reason to have an interface is to be able to add other concepts in the future, for example, fault tolerance.
- It was asked if Any_Task_State could be an access to an interface.
- An alternative was proposed to do the type Task_State a tagged private type and provide get and set methods for the attributes.
- It was suggested to add a Task_ID attribute to the state, with only a get operation.
- It was discussed where to put the initial priority. Suggestions to put it in the creation of the task, or to pass it as a discriminant (with the deadline) in order to avoid set operations (for initial priority and initial relative deadline). It was stated that having dynamic priorities is more useful, so finally set and get methods are needed.
- It was addressed what happened if a change on the relative deadline occurs, and when does it take effect. It was suggested to be at the next release of the task.
- Mode changes were also discussed with the meaning of having a task executing a different code for each mode and with different parameters (deadline, priority, etc.) for each mode. Solutions proposed were: derive a root type from Task_State with a collection of modes, or an array of Task_State so as to be able to change from periodic to sporadic for example, or allow a different task for each mode. The mode change issue is by itself very complex, so it was decided to focus the discussion on other issues, and leave the mode change aside, as future work.

At that point, the discussion turned onto the release mechanisms proposed in the framework [1]. The release mechanism is based on two operations Wait_For_Next_Release and Inform_Of (a deadline miss, an overrun, or both), and it is thought to build real-time task patterns using the select-then-abort statement. It was suggested to create a general abstract mechanism for notification. After a long discussion on that issue, it was agreed to implement the notification object (which consists of a set of events) with a synchronized interface as follows:

```
type Notification_Object
     is synchronized interface;
procedure Add  (N: Notification_Object;
                E: Event_Ptr) is abstract;
procedure Trigger  (N: Notification_Object;
                E: Event_Ptr) is abstract;
procedure Wait (N: Notification_Object;
                E: out Event_Ptr) is abstract;
-- Wait should be implemented with an entry
-- A function to delete events is also necessary
```

where the event is implemented as a tagged type without operations:

```
type Event_Kind is tagged with null record;
type Event_Ptr is access Event_Kind'class;
```

So it is possible to create deadline miss, overrun or other events by extending Event_Kind.

It was discussed whether the parameter for Wait should be an array of Event_Ptr, but it was decided that it was simpler to notify a single event each time Wait was called.

Another issue is how often the set of events to be notified is changed. If the set changes often, the Add operation would be eliminated and the set of events would be passed to the Wait operation at each call. It was considered however that the set of events is rather static and therefore separating the Add operation from the Wait operation is more convenient.

In addition, it would be necessary for the release mechanism to create an operation to add a reference to a notification object.

After finishing the discussion on the framework [1] and coming back to the framework proposed for Ravenscar [3], it was identified that the first one did not have the ability to pass parameters to sporadic tasks while the latter had. Tullio was asked to contribute to the framework [1] by integrating this ability.

6 Conclusions

The main conclusion is that some progress had been made but much more work is needed. The chair of the session suggested to propose a work plan:

- Alan Burns proposed to address the collaborative work needed to enhance a common framework via ARTIST meetings.
- Andy Wellings proposed some specific plans:
  - To change the notification object.
  - To work a little bit more on the framework before organizing a meeting to show the progress.
  - To ask for cooperation in specific topics of the implementation.

References

[1] Wellings, A.J., Burns, A. *A Framework for Real-Time Utilities for Ada 2005.* (this issue).

[2] Burns, A., Wellings, A.J. *Programming Execution-Time Servers in Ada 2005.* (this issue).

[3] Pulido, J., de la Puente, J.A., Bordin, M., Vardanega, T., Hugues, J. *Ada 2005 Code Patterns for Metamodel-Based Code Generation.* (this issue).

A Framework for Real-Time Utilities for Ada 2005*

A.J. Wellings, A. Burns

Department of Computer Science, University of York, UK; email: {andy,burns}@cs.york.ac.uk

Abstract

Modern large real-time systems are becoming more complex. Whilst Ada 2005 provides a comprehensive set of programming mechanisms that allow these systems to be implemented, the abstractions are low level. This paper argues that there is a need for a standardised library of real-time utilities that address common real-time problems

1 Introduction

Ada has comprehensive support for priority-based real-time systems. The approach has been to provide a set of low-level mechanisms that enable the programmer to construct systems solving common real-time problems. Whilst eminently flexible, this approach requires the programmer to re-implement common paradigms in each new system. In the past, these structures have been quite straightforward, perhaps just involving simply periodic or sporadic tasks communicating via protected data. However, modern large real-time systems are much more complex and include hard, soft and non real-time components. The resultingparadigmsare similarly more involved, and require activities like deadline miss detection, CPU budget overrun detection, the sharing of CPU budgets between aperiodic threads etc. Ada 2005 has responded admirably, expanding its set of low-level mechanisms. However, the common problems are now much more complex, and it is no longer appropriate to require the programmer to reconstruct the algorithms in each new system. Instead, what is required is a library of reusable real-time utilities; indeed, ideally such a library should become a de facto secondary standard – perhaps in the same way that Java has developed a set of concurrency utilities over the years that have now been incorporated into the Java 1.5 distribution.

The goal of this paper is to initiate a discussion in the Ada community to both confirm the need for a library of common real-time utilities and to propose (as a starting point) a framework for their construction. In section 2, an overview of the framework is given that also serves to indicate the scope of the utilities presented in this paper. Section 3 then presents the detailed design of the framework and shows how it can be used. Conclusions are drawn in section 4.

* An extended version of this paper also appeared in the proceedings of the Ada Europe 2007 conference. This work has been undertaken within the context of the EU ARTIST2 project.

2 Real-Time Utilities – Framework Overview

In the field of real-time programming, real-time tasks are often classified as being periodic, sporadic or aperiodic. Simple real-time periodic tasks are easy to program but once more complicated ones are needed (such as those that detect deadline misses, execution time (budget) overruns, minimum inter-arrival violations etc), the paradigms become more complex. Hence, there is a need to package up some of these and provide them as real-time tasking utilities.

A programmer wanting to use a real-time tasking abstraction will want to indicate (for example):

- whether the abstraction is periodic, sporadic or aperiodic (each task is "released" in response to a release event, which is usually time triggered for periodic tasks and event triggered for sporadic and aperiodic tasks);
- in the event of a deadline miss, either to terminate the current release of the task or to simply inform the program that this event has occurred (in which case, the programmer can choose to react or ignore the event);
- in the event of an execution time overrun, either to terminate the current release of the task or to simply inform the program that this event has occurred (in which case, the program can choose to react or ignore the event);
- whether a task is associated with an execution-time server that can limit the amount of CPU-time it receives.
- whether a task can operate in one or more modes, and if so, the nature of the mode change.

This section illustrates how real-time task abstractions, supporting some of these variations, can be developed in Ada2005.

The approach that has been taken is to divide the support for the tasks into four components.

1. The functionality of the task – this is encapsulated by the `Real_Time_Task_State` package. Its goal is to define a structure for the application code of the tasks. It is here that code is provided: to execute on each release of the task, to execute when deadline misses

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

occur and when execution time overruns occurs. In this paper, it is assumed that the task only wishes to operate in a single mode.

2. The mechanisms need to control the release of the real-time tasks and to detect the deadline misses and execution time overruns – this is encapsulated in the Release_Mechanisms package. Each mechanism is implemented using a combinations of protected objects and the new Ada 2005 timer and execution time control features.
3. The various idioms of how the task should respond to deadline misses and execution time overruns – this is encapsulated in the Real_Time_Tasks package. It is here that the actual Ada tasks are provided.
4. The mechanisms needed to encapsulate subsystems and ensure that they are only given a fixed amount of the CPU resource (often called temporal firewalling) – this is the responsibility of the Execution_Servers package. This paper focuses on using these mechanisms to support aperiodic task execution.

Figure 1 illustrates the top level packages that make up the abstractions. The details are discussed in the following section.

3 Framework Design

This section describes the details of the design of the framework introduced in Section 2. It assumes fixed priority scheduling and that the deadlines of allperiodic tasks are less than or equal to their associated periods.

3.1 RealTime Task State

First, it is necessary to provide a structure within which the programmer can express the code that the real-time task wishes to execute, along with its associated state variables. This is achieved, in the usual Ada object-oriented fashion, by defining the state within a tagged type, and providing operations to execute on the state. The following package shows the state and operations that all real-time tasks need.

```
--with and use clauses omitted
package Real_Time_Task_State is
  type Task_State is abstract tagged record
    Relative_Deadline : Time_Span;
    Execution_Time : Time_Span := Time_Span_Last;
    Pri : Priority := Default_Priority;
  end record;
  procedure Initialize(S: in out Task_State)
             is abstract;
   procedure Code(S: in out Task_State) is abstract;
   procedure Deadline_Miss(S: in out Task_State)
             is null;
   procedure Overrun(S: in out Task_State) is null;
   type Any_Task_State is access all Task_State'Class;
end Real_Time_Task_State;
```

Every real-time task has a deadline, an execution time and a priority. Here, these fields are made public, but they could have just as well been made private and procedures to 'get' and 'set' them provided. No additional assumptions

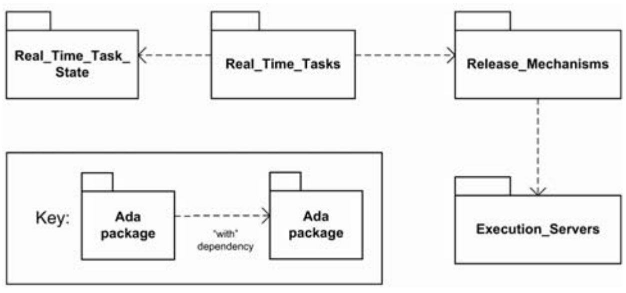


Figure 1 Top Level Packages

have been made about the values of these attributes. For example, the execution time could be worst-case or average.

The operations to be performed on a task's state are represented by the four procedures:

- Initialize –this code is used to initialize the real-time task's state when the task is created;
- Code – this is the code that is executed on each release of the task;
- Deadline_Miss – this is the code that is executed if a deadline is missed.
- Overrun – this is the code that is executed if an execution time overrun occurs.

Note, all real-time code must provide the Initialize and the Code procedures. There are default null actions on a missed deadline and on an execution time overrun.

Child packages of Real Time Task State provide support for periodic, aperiodic and sporadic task execution (as illustrated in Figure 2).

A periodic task's state includes that of a real-time task with the addition of its period of execution. In other words, it has regular time-triggered releases.

```
package Real_Time_Task_State.Periodic is
  type Periodic_Task_State is abstract new Task_State
     with record Period : Time_Span; end record;
  procedure Initialize(S: in out Periodic_Task_State)
              is abstract;
  procedure Code(S: in out Periodic_Task_State)
              is abstract;
  type Any_Periodic_Task_State is access all
        Periodic_Task_State'Class;
end Real_Time_Task_State.Periodic;
```

There is more than one model of a sporadic task; here, it is assumed that the task must have an enforced minimum inter-arrival time between releases (another approach would be to enforce a maximum arrival frequency). Hence, the state includes this value.

```
package Real_Time_Task_State.Sporadic is
  type Sporadic_Task_State is abstract new Task_State
      with record MIT : Time_Span; end record;
  procedure Initialize(S: in out Sporadic_Task_State)
              is abstract;
```

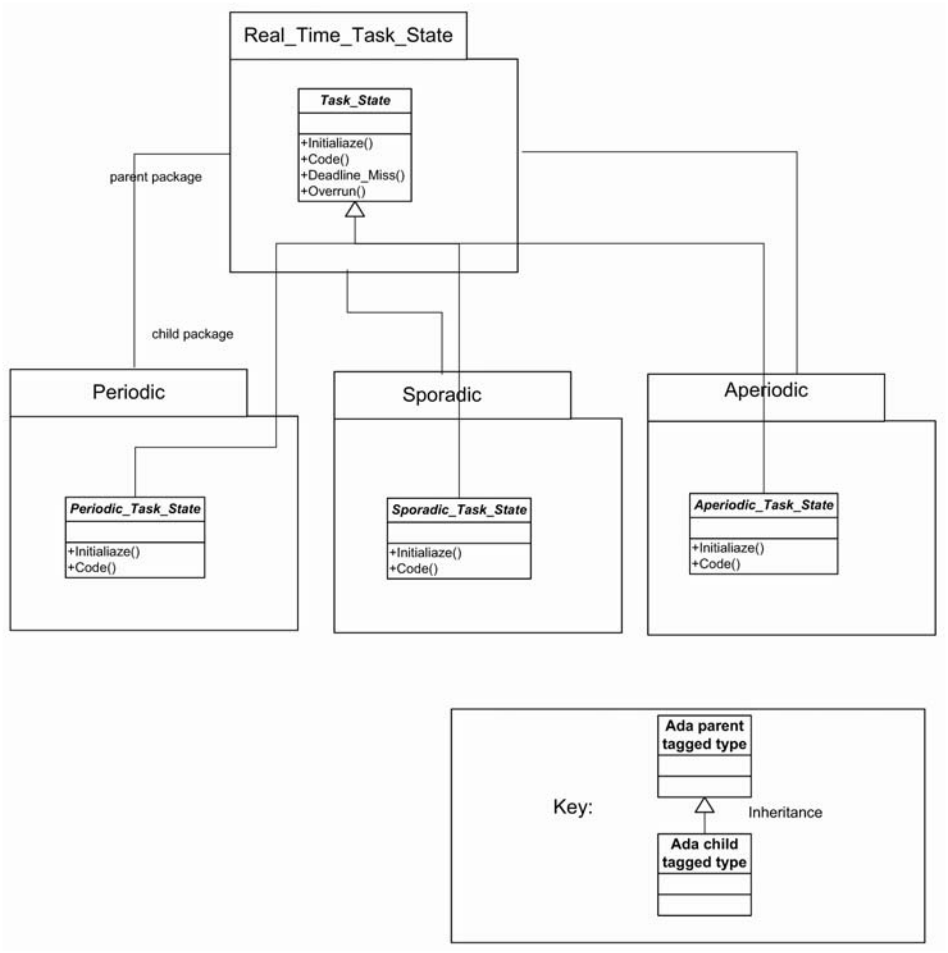


Figure 2 Task States

```
  procedure Code(S: in out Sporadic_Task_State)
           is abstract;
  type Any_Sporadic_Task_State is access all
        Sporadic_Task_State'Class;
end Real_Time_Task_State.Sporadic;
```

The state for aperiodic tasks has no new fields over the normal Task_State, but for uniformity, a new type can be created.

Application real-time tasks choose the appropriate real-time state to extend, and add their own state variables. For example, the following shows the application code to be used with the declaration of a periodic real-time task that is not interested in any missed deadlines or execution-time overruns.

```
type My_State is new Periodic_Task_State with
record
  --state variables
end record;
procedure Initialize(S: in out My_State);
procedure Code(S: in out My_State);

Example_State: aliased My_State := (
           Pri=> System.Default_Priority + 1);
```

3.2 Real-Time Task Mechanism

Real-time tasks can be released by the passage of time or via a software/hardware event. The following package (Release_Mechanisms) provides the common interfaces for all mechanisms (illustrated in Figure 3).

The root of the interface hierarchy (Release_Mechanisms) simply supports the facility for a real-time task to wait for notification of its next release to occur (be it a time or an event triggered release). Release_Mechanism_With_-Deadline_Miss is provided for the case where the real-time task wishes to be informed when it has missed a deadline. Similarly, Release_Mechanism_With_Overrun is provided for the case where the real-time task wishes to be informed when it has overrun its execution time. Finally, Release_Mechanism_With_Deadline_Miss_And_Overrun allows both detection of deadline misses and execution time overruns. The Ada code is shown below for some of the above.

```
package Release_Mechanisms is
  type Release_Mechanism is synchronized interface;
  procedure Wait_For_Next_Release(R: in out
          Release_Mechanism) is abstract;
  type Any_Release_Mechanism is access all
        Release_Mechanism'Class;
```

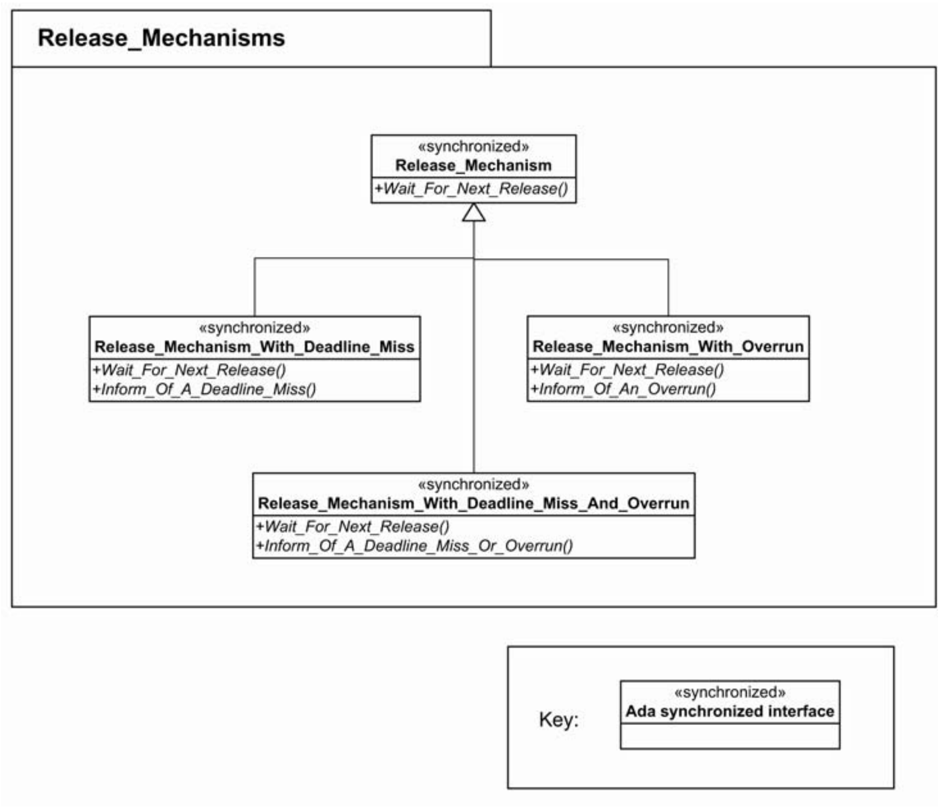


Figure 3 Release Mechanim Interfaces

```
  type Release_Mechanism_With_Deadline_Miss is
      synchronized interface and Release_Mechanism;
  procedure Wait_For_Next_Release(R : in out
      Release_Mechanism_With_Deadline_Miss)
      is abstract;
  procedure Inform_Of_A_Deadline_Miss(R : in out
      Release_Mechanism_With_Deadline_Miss)
      is abstract;
  type Any_Release_Mechanism_With_Deadline_Miss
      is access all
      Release_Mechanism_With_Deadline_Miss'Class;
  ...
end Release_Mechanisms;
```

Child packages provide the actual release mechanisms. For example, Figure 4 shows the mechanisms for periodic real-time tasks.

For example, the following shows the structure of a protected type that implements a periodic release mechanisms. The period of the task is obtained from the task's state, which is passed in as an access discriminant. The application code simply calls the Wait_For_Next_-Release entry.

```
--with and use clauses omitted
package Release_Mechanisms.Periodic is
  protected type Periodic_Release(
          S: Any_Periodic_Task_State) is
          new Release_Mechanism with
    entry Wait_For_Next_Release;
    pragma Priority(System.Interrupt_Priority'Last);
  private
    ...
  end Periodic_Release;
end Release_Mechanisms.Periodic;
```

Another protected type can support deadline miss detection

```
protected type Periodic_Release_With_DM(
      S: Any_Periodic_Task_State;
      Termination : Boolean) is
      new Release_Mechanism_With_Deadline_Miss with
    entry Wait_For_Next_Release;
    entry Inform_Of_A_Deadline_Miss;
    pragma Priority(System.Interrupt_Priority'Last);
private
  ...
end Periodic_Release_With_DM;
```

Here, a boolean indicates whether the application requires notification or termination of a deadline miss. If termination is required, the Inform_Of_A_Deadline_Miss entry can be used by theframeworkin a select-then-abort statement.

3.3 Aperiodic release mechanisms and execution servers

The final type of release mechanism is that for handling aperiodic releases. Typically, the CPU time allocated to aperiodic tasks must be constrained as they potentially can have unbounded resource requirements. The Ada 2005 group budget facility can be used to implement the various approaches.

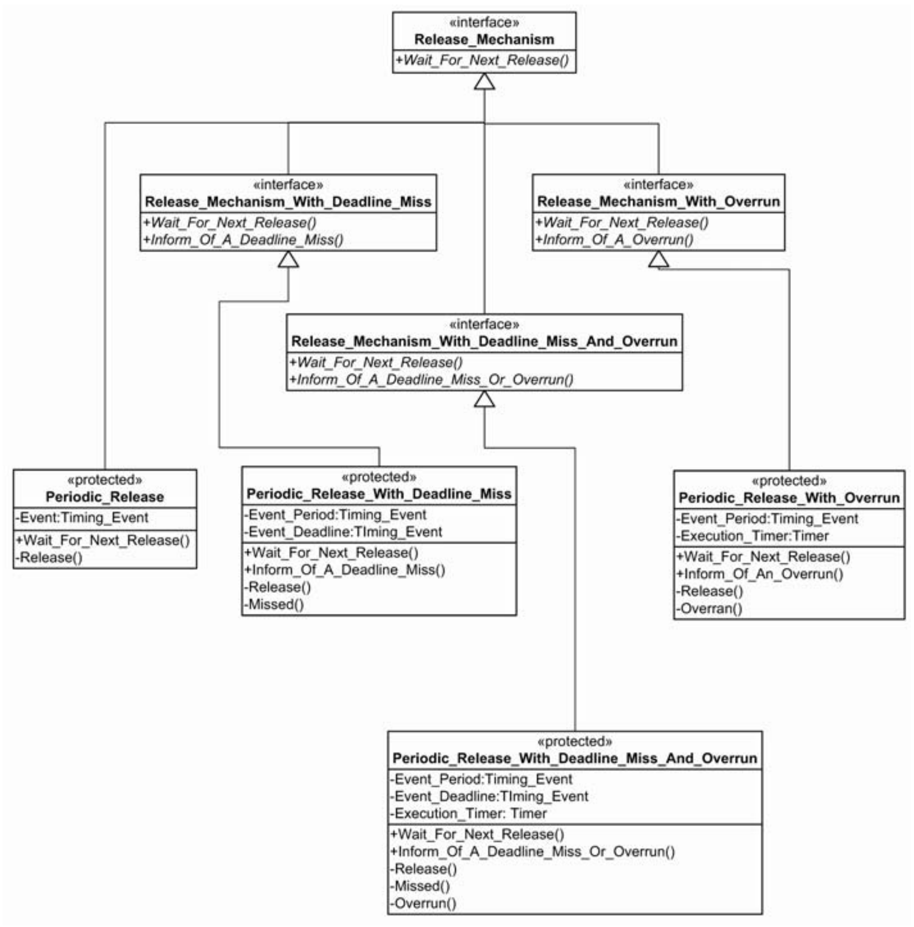


Figure 4 Release Mechanism Classes

Before the release mechanism can be programmed, it is necessary to consider how it interacts with the execution severs. This paper will only consider periodic execution servers. The following package specification defines the common interface.

```
package Execution_Servers is
  type Server_Parameters is tagged record
    Period : Time_Span; Budget : Time_Span;
  end record;

  type Execution_Server is synchronized interface;
  procedure Register(ES: in out Execution_Server;
              T : Task_Id) is abstract;
  ...
  type Any_Execution_Server is access
              all Execution_Server'Class;
end Execution_Servers;
```

All servers have parameters that determine the servers characteristics. They include:

- the budget – how much CPU time has been allocated to the server;
- the period – this relates to how often the server's budget is replenished.

Two of the main servers found in the literature (the deferrable [1] and sporadic servers [2]) also require their clients to have foreground and background priorities, but in the general case this may not be the situation. Some servers suspend their clients when their execution time expires, and other servers allow their clients to have different priorities.

All execution servers require their clients to register. Here, any task can register any other task. Some types of execution servers will also want to know when the client tasks are executable. These periods of execution are called sessions. The associated procedures have default null values.

To facilitate the use of execution servers, it is necessary to modify the release mechanism to allow the actual server to be passed as a discriminant. The approach is illustrated by considering aperiodic releases (although, the same approach could be applied to the periodic or sporadic release mechanisms).

```
package Release_Mechanisms.Aperiodic is
  protected type Aperiodic_Release(
                S: Any_Aperiodic_Task_State;
                ES: Any_Execution_Server) is
    new Release_Mechanism with
      entry Wait_For_Next_Release;
      procedure Release;
      pragma Priority(Interrupt_Priority'Last);
   private
      ...
   end Aperiodic_Release;
end Release_Mechanisms.Aperiodic;
```

3.4 RealTime Tasks

The Real Time Tasks package provides the code that integrates the task states with the release mechanisms to provide the required application abstraction. Several task types are shown in the following package specification.

```
-- with and use clauses omitted
package Real_Time_Tasks is
   task type Simple_RT_Task(
            S : Any_Task_State;
            R : Any_Release_Mechanism;
            Init_Prio : Priority) is
       pragma Priority(Init_Prio);
   end Simple_RT_Task;
   task type RT_Task_With_Deadline_Termination(
            S : Any_Task_State;
            R : Any_Release_Mechanism_With_DM;
            Init_Prio : Priority) is
     pragma Priority(Init_Prio);
   end RT_Task_With_Deadline_Termination;
   ...;
end Real_Time_Tasks;
```

Others can be designed; for example, for the cases where the current release of the task must be immediately terminated if the execution time is exceeded.

Note that the priority at which the task activates can be given as a discriminant (this can be changed in the application initialization code for the execution phase of the task).

Note also that if the Simple_RT_Task is used with one of the release mechanisms that support deadline miss or execution-time overrun detection, it should set the Termination discriminant to false. This will allow the task to be notified using a direct call to the Deadline_Miss_-and_Overrun operations via the Task_State.

The body of this package shows the various structures. Note the use of the 'select-then-abort' statement to achieve the termination semantics.

```
package body Real_Time_Tasks is
   task body Simple_RT_Task is
   begin
     S.Initialize;
     loop
       R.Wait_For_Next_Release;
       S.code;
     end loop;
   end Simple_RT_Task;
   task body RT_Task_With_Deadline_ Termination is
   begin
     S.Initialize;
     loop
       R.Wait_For_Next_Release;
       select
         R.Inform_Of_A_Deadline_Miss; S.Deadline_Miss;
       then abort
         S.code;
       end select;
     end loop;
   end RT_Task_With_Deadline_ Termination;
   -- similar structures for the other task types
end Real_Time_Tasks;
```

3.5 A simple example

Consider a simple example of two identical periodic tasks that wish to detect deadline misses: one requires termination semantics, the other just wishes to be notified. First, the application code is defined

```
type My_State is new Periodic_Task_State with record
  I : Integer;
end record;

procedure Initialize(S: in out My_State);
procedure Code(S: in out My_State);
procedure Deadline_Miss(S: in out My_State);

Example_State1: aliased My_State;
Example_State2: aliased My_State;

Releaser1 : aliased Periodic_Release_With_DM(
      Example_State1'Access, Termination => True);
Releaser2 : aliased Periodic_Release_With_DM(
      Example_State2'Access,
      Termination => False);
```

In the above, two instances of the state are created, one for each real-time task. There are twoprotected objects for the release mechanisms: Release1 supportsthe termination model, and Release2 supportsthe notification model

Now, the real-time tasks can be declared:

```
T1 : RT_Task_With_Deadline_Termination(
           Example_State1'Access, Releaser1'Access,
           Default_Priority);
T2 : Simple_RT_Task (
          Example_State2'Access, Releaser2'Access,
          Default_Priority);
```

Here, T1 uses the real-time task type that supports the termination semantics, and T2 uses the simple real-time task type.

For completeness, the code of the tasks are given (they are identical, in this example and simply manipulate an integer state variable).

```
procedure Initialize(S: in out My_State) is
begin
  S.I := 2; S.Pri := Default_Priority + 2;
  S.Relative_Deadline := To_Time_Span(0.1);
end Initialize;
```

```
procedure Code(S: in out My_State) is
begin
  S.I := S.I * 2;
end Code;

procedure Deadline_Miss(S: in out My_State) is
begin
  S.I := 2;
end Deadline_Miss;
```

The body initializes the state, sets the priority and the deadline; it then squares the state value on each periodic release. On a deadline miss, the value is reset to2.

4 Conclusions

The Ada designers have learned well the lesson of trying to support too higher level abstractions for real-time programming, that was evident in Ada 83. That version was heavily criticised. Ada 95 responded to those criticisms and is an efficient language for high-reliable long-lived real-time applications. Ada 2005 has continued the recovery, and the language now provides a comprehensive set of mechanisms that can support the development of modern large real-time systems.

However, the complexity of modern real-time systems means that there is now the need to provide high-level real-time programming abstractions in the form of standardised library utilities. The goal of this paper has been to start the debate on how the Ada community should respond to this challenge. The initial focus has been on the construction of a framework that allows the provision of real-time tasking utilities. The framework has been implemented using the evolving Ada 2005 compiler from AdaCore along with a local simulation facilities for the new Ada 2005 real-time mechanisms.

References

[1] J.P. Lehoczky, L. Sha, and J.K. Strosnider. *Enhanced aperiodic responsiveness in a hard real-time environment*. In 8th IEEE RTSS, pages 261–270, 1987.

[2] B. Sprunt, J. Lehoczky, and L. Sha. *Exploiting unused periodic time for aperiodic service using the extended priority exchange algorithm*. In 9th IEEE RTSS, pages 251–258, 1988.

Programming Execution-Time Servers in Ada 2005

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

Abstract

Much of the research on scheduling schemes is prevented from being used in practice by the lack of implementation sthat provide the necessary abstractions. An example of this lack of provision is the support of execution-time servers, these important building blocks are not generally available to the system developer. In this paper, we show how new Ada 2005 mechanisms can be used to construct various execution-time servers. We also outline the different server types that could form part of a library of real-time utilities for Ada.

1 Introduction

One of the key building blocks for delivering flexible scheduling is the use of execution-time servers [10, 11, 9, 2, 1] – we will refer to these as just servers in the following discussions. Server are in some senses virtual processors, they provide their clients with a budget that has a defined ‘shelf life’ and a means of replenishing the budget in a predictable and analysable way. The servers collectively must manage their budgets (i.e. allow the budget to be available to clients), whilst the clients must ensure that the budget is sufficient for their needs. A number of papers have addressed the scheduling issues associated with server-based systems. In this paper we are concerned with programming servers.

The paper is organised as follows. First two examples of the expressive power of Ada 2005 are illustrated: the programming of the Deferrable and then the Sporadic server. The code in these examples comes from a recent publication [6]. This is followed in Section 4 by a discussion of the various properties servers can have. This discussion is intended to lead to the definition of reusable real-time utilities that could form part of a library or secondary standard [12].

2 Deferrable Server

In this section we illustrate how the Deferrable Server can be constructed. Space considerations mean that the Ada 2005 facilities are not described. Here, the server has a fixed priority (i.e. all client tasks execute with the same priority), and when the budget is exhausted the tasks are moved to a background priority. A client task first registers with its server, after that, in this simple example, it has no direct interactions with the server. The server itself needs a high interrupt priority level as it handles timing events.

```
-- with clauses are omitted for brevity
package Deferrable is
  type Deferrable_Server_Parameters is
  record
    Period : Time_Span;
    Budget : Time_Span;
    Foreground_Pri : Priority;
    Background_Pri : Priority;
  end record;

  protected type Deferrable_Server(
       Params : access   Deferrable_Server_Parameters)
     procedure Register(T : Task_Id := Current_Task);
    pragma Priority(System.Priority'Last);
  private
    procedure Timer_Handler(E : in out Timing_Event);
      -- fired when replenishment is due
    procedure Group_Handler(G : in out Group_Budget);
      -- fired when budget exhausted
    T_Event : Timing_Event;
    G_Budget : Group_Budget;
    First : Boolean := True;
  end Deferrable_Server;
end Deferrable;
```

A single timing event is used to replenish the budget at regular timing intervals. A group budget is employed to take the necessary actions when the current budget is exhausted. In the following code the first client task to register sets up the server; this could have been done by a separate routine called when the server is created.

```
package body Deferrable is
  protected body Deferrable_Server is
    procedure Register(T : Task_Id := Current_Task) is
    begin
      if First then
        First := False;
        G_Budget.Add(Params.Budget);
        T_Event.Set_Handler(Params.Period,
                            Timer_Handler'Access);
        G_Budget.Set_Handler(Group_Handler'Access);
      end if;
      Add_Task(G_Budget, T);
      if G_Budget.Budget_Has_Expired then
        Set_Priority(Params.Background_Pri);
        -- sets client task to background priority
      else
        Set_Priority(Params.Foreground_Pri);
        -- sets client task to servers 'priority'
```

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

```
      end if;
    end Register;
    procedure Timer_Handler(E : in out Timing_Event) is
      T_Array : Task_Array := G_Budget.Members;
    begin
      G_Budget.Replenish(Params.Budget);
      for ID in T_Array'Range loop
        Set_Priority(Params.Foreground_Pri,T_Array(ID));
      end loop;
      E.Set_Handler(Params.Period,
                    Timer_Handler'Access);
    end Timer_Handler;

    procedure Group_Handler(
                    G : in out Group_Budget) is
      T_Array : Task_Array := G_Budget.Members;
    begin
      if G_Budget.Budget_Has_Expired then
        -- test needed to cover race condition
        -- with timer handler
        for ID in T_Array'Range loop
          Set_Priority(Params.Background_Pri,
          T_Array(ID));
        end loop;
      end if;
    end Group_Handler;
  end Deferrable_Server;
end Deferrable;
```

3 Banded Sporadic Server

To programme the Sporadic Server also requires the use of timing events and a group budget. In the following, tasks are suspended if there is no budget, and a finite set of reused timing budgets is employed. If this number if exhausted, the POSIX scheme of concatenating two replenishments into one is employed [8].

The client is assumed to request the use of the budget by bracketing (non-blocking) application code in the following way:

```
BSS.Register;
...
loop
  BSS.Start_Session;
  -- where BSS is of the server type below

  -- non-blocking code
  BSS.Complete_Session;
end loop;
```

The Sporadic Server allows its clients to preempt each other by supporting a range of priority levels (hence the term *banded sporadic server*). All clients must have a priority within this range. With the sporadic server a client that arrives at time t and uses budget b, results in the replenishment of b at time t+T –where T is the period of the server. To program this, the timing event must know how much budget to return. This is accommodated by extending the timing event type.

```
package Banded_Sporadic is
  Priority_Out_Of_Range : exception;
  Already_Member_Of_A_Group_Budget: exception;
  type Banded_Sporadic_Server_Parameters is record
    Period : Time_Span;
    Budget : Time_Span;
    Low_Priority : Priority; -- of the band
    High_Priority : Priority; -- of the band
  end record;

  type Budget_Event is new Timing_Event with
  record
    Bud : Time_Span;
  end record;

  type Bud_Event is access Budget_Event;
  type Bud_Events is array(Natural range <>)
         of Budget_Event;

  protected type Banded_Sporadic_Server
        (Params : access
                  Banded_Sporadic_Server_Parameters;
         No_Timing_Events : Positive) is
    pragma Interrupt_Priority (Interrupt_Priority'Last);
    procedure Start_Session(
         T : Task_Id := Current_Task);
    procedure Complete_Session(
         T : Task_Id := Current_Task);
    procedure Register(T : Task_Id := Current_Task);
  private
    procedure Timer_Handler(E : in out Timing_Event);
    procedure Group_Handler(
         G : in out Group_Budget);
    G_Budget : Group_Budget;
    B_Events : Bud_Events(1 .. No_Timing_Events);
    Next : Natural := No_Timing_Events;
    Number : Natural := 0;
    Start_Budget : Time_Span;
    Release_Time : Time;
    Tasks_Executing : Natural := 0;
    First : Boolean := True;
  end Banded_Sporadic_Server;
end Execution_Servers.Banded_Sporadic;
```

The only real complexity in this code comes from the reuse of a finite collection of timing events. The procedure Set_Timing_Events is used to manage this.

```
with Ada.Asynchronous_Task_Control;
use Ada.Asynchronous_Task_Control;
package body Execution_Servers.Banded_Sporadic is
  protected body Banded_Sporadic_Server is
    procedure Set_Timing_Events(B : Time_Span;
                                T : Time) is
    begin
      if Number < No_Timing_Events then
        Next := Next mod No_Timing_Events + 1;
        B_Events(Next).Bud := B;
        B_Events(Next).Set_Handler(T,
        Timer_Handler'Access);
        Number := Number + 1;
      else
      B_Events(Next).Bud := B_Events(Next).Bud + B;
      B_Events(Next).Set_Handler(T,
      Timer_Handler'Access);
    end if;
  end Set_Timing_Events;
  procedure Register(T : Task_Id := Current_Task) is
  begin
    if not (Get_Priority(T) in Params.Low_Priority ..
                   Params.High_Priority) then
      raise Priority_Out_Of_Range;
    end if;
    if First then
      First := False;
      G_Budget.Add(Params.Budget);
      G_Budget.Set_Handler(Group_Handler'Access);
    end if;
    G_Budget.Add_Task(T);
  exception
    when Group_Budget_Error =>
      raise Already_Member_Of_A_Group_Budget;
  end Register;

  procedure Start_Session(T : Task_Id) is
  begin
    if Tasks_Executing = 0 then
      Release_Time := Clock;
      Start_Budget := G_Budget.Budget_Remaining;
    end if;
    Tasks_Executing := Tasks_Executing + 1;
    if G_Budget.Budget_Has_Expired then
      Hold(T);
    end if;
  end Start_Session;

  procedure Complete_Session(T : Task_Id) is
  begin
    -- work out how much budget used, construct
    -- timing event and set the handler
    Tasks_Executing := Tasks_Executing - 1;
    if Tasks_Executing = 0 then
      Set_Timing_Events(
            Start_Budget - G_Budget.Budget_Remaining,
            Release_Time + Params.Period);
    end if;
  end Complete_Session;

  procedure Timer_Handler(
              E : in out Timing_Event) is
    Bud : Time_Span;
    T_Array : Task_Array := G_Budget.Members;
  begin
    Number := Number - 1;
    Bud := Budget_Event(Timing_Event'Class(E)).Bud;
    if G_Budget.Budget_Has_Expired then
      G_Budget.Replenish(Bud);
      for I in T_Array'range loop
        Continue(T_Array(I));
      end loop;
      Release_Time := Clock;
      Start_Budget := Bud;
    else
      G_Budget.Add(Bud);
      Start_Budget := Start_Budget+Bud;
    end if;
  end Timer_Handler;

  procedure Group_Handler(
              G : in out Group_Budget) is
    T_Array : Task_Array := G_Budget.Members;
  begin
    if G_Budget.Budget_Has_Expired then
      -- a replenish event required for the
      -- budget used so far
      Set_Timing_Events(Start_Budget, Release_Time +
                              Params.Period);
      for I in T_Array'range loop
        Hold(T_Array(I));
      end loop;
    end if;
  end Group_Handler;
 end Banded_Sporadic_Server;
end Execution_Servers.Banded_Sporadic;
```

A replenishment event is set up either when the current budget is exhausted or, at the end of a client's session, if there are no longer any active clients.

4 Different Server Characteristics

In this section a number of server characteristics are considered. In many instances these characteristics are orthogonal and hence gives rise to a wide range of possible strucures. In all of this discussion fixed priority scheduling is assumed. The use of servers and other scheduling policies such as EDF is not considered in this paper.

4.1 Dispatching

Here two characteristics are identified: concurrency within the server, and the behaviour of the tasks when the server capacity is exhausted. With the former there are three possibilities:

1. All client tasks have the same priority and hence their use of the server's budget is serialised.
2. Client task have distinct priorities but the range of priorities for each server is disjoint. Client tasks can now preempt each other and thereby exhibit a more responsive behaviour.
3. Client task have distinct priorities, and there are no constraints on the priorities. This is the general model supported by Java's (RTSJ) processing groups [4] – but the resulting system is not easily amenable to scheduling analysis [5].

The code for the Deferrable and Sporadic servers have illustrated the first two of these schemes. They have also used alternative policies for dealing with tasks when there is no budget available:

1. Run the client tasks at a background (low) priority.
2. Suspend the tasks.

The suspension of the tasks can make certain aspects of the implementation easier (as the client tasks cannot have executed during the time the server has no capacity). Also with a large system with many servers, there may be little likelihood of spare capacity been available at the background level, and hence the use of suspension is not as inefficient as it might seem.

4.2 Binding

Here a task coordinates its release with the replenishment of the server (either a Deferrable or Periodic server – see below). The task is described as being *bound* to the server. The advantage of this scheme is that it makes the scheduling analysis of these bound tasks less pessimistic [7]. One means of achieving this binding is to introduce into the server a start session entry that the client task calls as their 'wait for next invocation' event.

```
entry Start_Session(T : Task_Id := Current_Task)
      when Released is
begin
  Released := Start_Session'Count > 0;
end Start_Session;
```

With the boolean barrier being set in the timing event handler:

```
Released := Start_Session'Count > 0;
```

4.3 Stop on Exhaustion of the Budget

The above illustrates a coordination between the replenishment of a budget and the release of a task. Another form of coordination is between the exhaustion of the budget and the performance of some 'non-terminating' algorithm that is 'stopped' when there is no longer any budget. To implement this, the server would have an entry (e.g. `Exhausted`) that has a barrier variable that is set to true when the group budget event is fired. The application code will call this entry from an ATC structure:

```
loop
  Some_Server.Start_Session;
  select
    Some_Server.Exhausted;
    -- Place result in some appropriate object
  then abort
    code
  end select;
end loop;
```

This server does not change the priority of the client when the budget is exhausted but a `Start_Session` entry is used that is only open when there is budget available. The two entries, `Exhausted` and `Start_Session` could easily be added to the Deferrable Server's specification.

4.4 Budget Sharing

There are a number of schemes described in the literature (eg. [3, 1]) that allow the budget in one server to be passed to another server if there are no local clients requiring service. A collection of fixed priority Deferrable Servers (for example) could be constructed so that each server always knows its 'neighbour'. This is the server with the next highest priority. If this neighbour had active clients they would be executing; as it isn't executing it must have either no such clients or have no current budget. The following scheme is based on the premise that when a budget is exhausted (i.e. the group budget event is fired) an attempt is made to pull capacity down from its neighbour. Only if this fails are the client tasks demoted. The neighbour will pass on a gift of its current budget if it has one – if not, it will attempt to pull down budget from its neighbour.

To implement this scheme, a new function is added to the interface of the Deferrable Server (`Extract`). The group handler can now be executed when either one of its own clients was active and the budget was exhausted or the remaining budget was gifted away. In the latter case no attempt is made to extract extra budget from the servers' neighbour. A new boolean variable Gift_Aid is used to distinguish between these two cases. The overall approach is sketched below.

```
package body Deferrable is
  protected body Deferrable_Server is
    procedure Register(T : Task_Id := Current_Task)
              is ...
    procedure Timer_Handler(E : in out Timing_Event)
              is ...
    procedure Group_Handler(
              G : in out Group_Budget) is
      T_Array : Task_Array := G_Budget.Members;
      Gift : Time_Span := Time_Span_Zero;
    begin
      if G_Budget.Budget_Has_Expired then
        if Gift_Aid then
          Gift := Time_Span_Zero;
        else
          Gift := Neighbour.Extract;
        end if;
        if Gift = Time_Span_Zero then
          for ID in T_Array'Range loop
            Set_Priority(Params.Background_Pri,
                         T_Array(ID));
          end loop;
        else
          G_Budget.Replenish(Gift);
        end if;
        Gift_Aid := False;
      end if;
    end Group_Handler;

    function Extract return Time_Span is
      Gift : Time_Span;
    begin
      if G.Budget.Budget_Has_Expired then
        return Neighbour.Extract;
        -- if highest priority server then no neighbour
        -- so return Time_Span_Zero
      end if;
      Gift_Aid := True;
      Gift := G_Budget.Budget_Remaining;
      G_Budget.Replenish{Time_Span_Zero);
      return Gift;
    end Extract;
  end Deferrable_Server;
end Deferrable;
```

Obviously the use of capacity sharing must meet the re-uirements of the application. A client may have to wait until the next replenishment in order to execute – as its server's capacity has been given away. Hence, it is best to use this technique with servers that have bound tasks (see section 4.2). If the technique is too extreme, the scheme can be restricted so that only a proportion of the server's capacity is gifted away.

Other kinds of capacity sharing are also possible. There is a need to characterise these and show how they can be implemented in Ada 2005.

4.5. Server Types

In addition to Deferrable and Sporadic servers there are a number of other schemes discussed in the literature. The main additional one is the Periodic Server. This behaves like a Deferrable Server except that its capacity is not preserved during the server's period; it is only available to clients who are ready to execute at the time the budget is replenished. This has the advantage that the Periodic Server behaves just like a periodic task - in terms of its impact on lower priority tasks and servers. This is not the case for a Deferrable Server that can have an increased impact due to its clients arriving late one period and early the next. There are two distinct means of providing the required behaviour for a Periodic Server:

1. Each server has a looping client that has the lowest priority of all users of the server but which executes a non-blocking busy loop. It will always be available to execute and will use up all the available capacity.
2. When there are no current clients left to execute, reduce the budget to zero. If budget sharing is to be employed then the remaining budget could be passed on by adding it to the available budget of another server.

The latter clearly is less wasteful and can be easily introduced by again using a start and end session interface. The server keeps a count of the number of active agents; when this value goes to zero the remaining budget is removed (or re-assigned).

5 Conclusion

The focus of this paper has been on the construction of server abstractions using the new facilities of Ada 2005. Simple servers such as the Deferrable Server are straightforward and need just a simple timing event and a group budget. The Sporadic Server by contrast is quite complicated and its implementation is a testament to the expressive power of the language.

One of the motivations of this paper is to start to define a collections of useful server abstractions, and to facilitate the use of these abstractions via a library of tested components, and/or a secondary standard.

Acknowledgements

The work reported in the paper is funded, in part, by the EU project ARTIST.

References

[1] G. Bernat, I. Broster, and A. Burns. *Rewriting history to exploit gain time*. In Proceedings Real-time Systems Symposium, pages 328–335, Lisbon, Portugal, 2004. Computer Society, IEEE.

[2] G. Bernat and A. Burns. *New results on fixed priority aperiodic servers*. In Proceedings 20th IEEE Real-Time Systems Symposium, pages 68–78, 1999.

[3] G. Bernat and A. Burns. *Multiple servers and capacity sharing for implementing flexible scheduling*. Real-Time Systems Journal, 22:49–75, 2002.

[4] G. Bollella, B. Brosgol, P. Dibble, S. Furr, J. Gosling, D. Hardin, and M. Turnbull. *The Real-Time Specification for Java*. Addison-Wesley, 2000.

[5] A. Burns and A. J. Wellings. *Processing group parameters and the real-time specification for java*. In On the Move to Meaningfull Internet Systems 2003: Workshop on Java Technologies for Real-Time and Embedded Systems, volume LNCS 2889, pages 360–370. Springer, 2003.

[6] A. Burns and A.J. Wellings. *Programming execution-time servers in ada 2005*. In Proceedings of the 27th IEEE Real- Time Systems Symposium, pages 47–56, 2006.

[7] R. Davis and A. Burns. *Hierarchical fixed priority preemptive scheduling*. In IEEE Real-Time Systems Symposium, pages 389–398, 2005.

[8] IEEE Std.1003.1c-1995. *Information Technology – Portable Operating System Interface (POSIX): Part 1 : System Application program interface (API) – Amendment 2: Threads Extension* [CLanguage], 1995.

[9] J.P. Lehoczky and S. Ramos-Thuel. *An optimal algorithm for scheduling soft-aperiodic tasks fixed-priority preemptive systems*. In Proceedings 13th IEEE Real-Time Systems Symposium, pages 110–123, 1992.

[10] J.P. Lehoczky, L. Sha, and J.K. Strosnider. *Enhanced aperiodic responsiveness in a hard real-time environment.* In Proceedings 8th IEEE Real-Time Systems Symposium, pages 261–270, 1987.

[11] B. Sprunt, L. Sha, and J. P. Lehoczky. *Aperiodic task scheduling for hard real-time systems*. Real-Time Systems, 1:27–69, 1989.

[12] A.J. Wellings and A. Burns. *A framework for real-time utilities for Ada 2005*. (this issue).

Ada 2005 Code Patterns for Metamodel-Based Code Generation[*]

José A. Pulido, Juan A. de la Puente

Universidad Politécnica de Madrid (UPM), Spain; email: {pulido,jpuente}@dit.upm.es

Jérôme Hugues

GET-Télécom Paris – LTCI-UMR 5141 CNRS (ENST), Paris, France; email: hugues@infres.enst.fr

Matteo Bordin, Tullio Vardanega

Università di Padova, Italy; email: {mbordin, tullio.vardanega}@math.unipd.it

Abstract

In this paper we discuss the issues we have explored, as part of the ASSERT project, in the definition and implementation of Ada coding patterns for the support of the automated code generation stage of a comprehensive approach to model-driven development for high-integrity systems.

1 Introduction

In the ASSERT project we explored the feasibility, practicality and performance of a Model-Driven development (MDA) approach targeted to high-integrity applications. A crucial element of our vision relied on the definition and exploratory implementation of a strategy for the automated transformation of the system model down to source code and to execution in a manner capable of warranting the preservation of the properties stipulated at model level.

To support system modeling we introduced two types of containers that are distinct for use but very strictly related when it comes to applying model transformation. One type of container, which we call "*application level container*" addresses functional modeling, where the level of expression ought to be abstracted away from the intricacies of hard real-time concurrency (much in keeping with the platform-independent model, PIM, of MDA). The other type of container, which we name "*virtual machine level container*", designates entities that exist at run time with an intended semantics that must be actively preserved by the execution platform. In the following we shall denote the former by "*APLC*" and the latter by "*VMLC*". We then use the term "*virtual machine*" (VM) to express the kind of execution platform we require to warrant property preservation in the trajectory from model to execution.

In our vision, designers operate on APLC only. Such containers however are not executable, while VMLC are. There must therefore exist some logic that transforms the former in the latter in a trustworthy manner, without any semantic distortion and with active preservation and enforcement of properties. Such a transformation takes the model from a higher level of abstraction to a lower one, closer to implementation and execution. Hence we regard it as a "vertical transformation", as opposed to other forms of model transformation in which the representation formalism changes while the level of abstraction need not. Our principle of vertical transformation rests on the postulate that the attributes set on APLC (which denote component interfaces and relations among them) consistently map to groups of interconnected VMLC endowed with the required interfaces.

The platform must be rigidly inflexible (as opposed to permissive) in hosting, executing and actively policing the run-time behaviour of the container entities that may legally exist at its level. In fact, our VM concept entails the following characteristics:

1. it is a run-time environment that only accepts and supports "legal" entities; the sole legal entities that may be propagated down from model transformation are VMLC; no other run-time entity is permitted to exist and no other can thus be assumed in the model
2. it provides run-time services that assist containers in actively preserving their designated properties; mechanisms and services of interest may for instance aid to:
    - accurately measure the actual execution time that can be attributed to individual threads of control
    - attach and replenish a monitored execution time budget to a thread, and then prompt an alarm (e.g.: an exception) when the thread should exceed its time budget
    - segregate threads into distinct groups, attaching a monitored budget to individual groups, to be handled in the same way as for threads
    - enforce the minimum inter-arrival time stipulated for sporadic threads

[*] This work has been funded in part by the IST Programme of the European Comission under project IST-004033 (ASSERT)

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

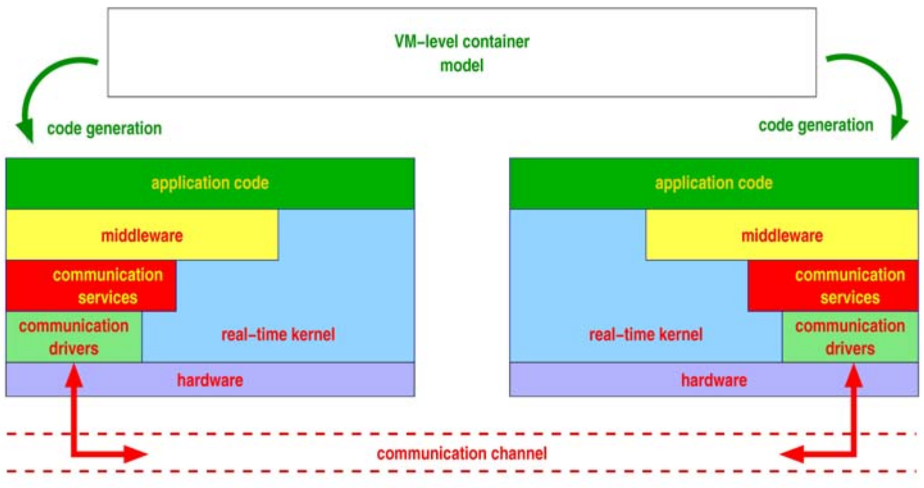


Figure 1 Top-level VM architecture

- build fault containment regions around individual threads and groups thereof
- attain distribution and replication transparency in inter-thread communication.

3. it is bound to a compilation system that only produces executable code for “legal” entities and rejects the non-conforming ones; run-time checks provided by the VM shall cover the extent of enforcement that cannot be exhaustively achieved at compile and link time
4. it realizes a concurrent computational model provably amenable to static analysis; the model must permit threads to interact with one another (directly, by some form of synchronization, and indirectly, by pre-emptive interference) in ways that do not incur non-determinism.

2 Virtual Machine Architecture

To date, we consider that the ASSERT VM should be constituted by the integration of the following key elements, shown in figure 1.

- *Real-Time kernel*: it assigns CPU time to threads, providing tasking and synchronization primitives. It also supports some mechanisms for device drivers to access the hardware elements of the targets;
- *Middleware*: it provides the necessary abstractions to support communication between nodes of the application. These abstractions are defined as part of the ASSERT Distribution model.
- *Communication subsystem*: it provides low-level support for inter-partition communications relying on the deterministic SpaceWire protocol [6];

All of those elements are individually defined in full compliance with both the Ravenscar profile and the restrictions set by the High-Integrity systems annex of Ada 2005 [11]. The key high-level restrictions that we placed on them are as follows:

2.1 Real-time kernel

The real-time kernel component of the ASSERT VM is an upgraded version of the Open Ravenscar Kernel (ORK) [2, 4]. It provides direct support for the Ravenscar profile [11, D.13] *and* includes the following Ada 2005 extensions:

- global timing events;
- execution-time clocks;
- system-wide (fallback) teask termination handler.

The kernel also supports execution-time timers and group budgets. Although not allowed in the Ravenscar profile, those mechanisms help enforce temporal separation between subsystems with possibly different levels of criticality, which is a strong requirement for the kind of on-board aerospace embedded systems envisaged in the ASSERT project. Execution-time timers and budgets are used to implement hierarchical scheduling of subsystems as a means to provide temporal isolation [9]. The kernel only allows one execution-time timer per task, as suggested in previous IRTAW discussions [5, 3].

The ASSERT VM kernel is integrated with the GNAT compilation system.

2.2 Middleware

The distribution layer of the ASSERT VM (the “middleware”) is placed at the interface between the application code and the underlying kernel. It maps high-level interactions onto calls to low-level primitives that support distribution in nearly-transparent manner. The subset of requirements of the Middleware that affects the kernel is well defined. Those that most impact the design of the Middleware component of the ASSERT VM are as follows:

- The middleware inherits and restricts the process and data models of the real-time kernel and it is therefore able to provide adequate support for timeliness and predictability of service.
- Distribution transparency: the ASSERT VM should rely on the process model and provide the infrastructure and constructs required to provide for

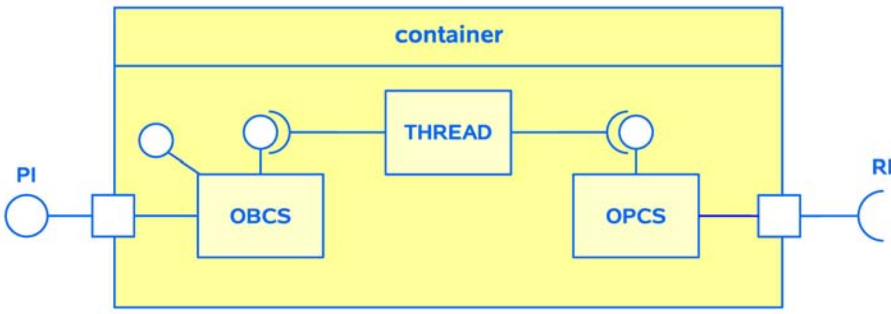


Figure 2 VM-level container meta-model

distribution transparency. As a result of this provision, the ASSERT designer should not need to modify source code to deploy different configurations of the system.

The communication services that interact with the physical interconnection might also impose some requirements to be fulfilled by the middleware. The analysis of its full impact is currently in progress, due to complete by Q3 2007. As an adaptation layer, the middleware shall limit its weight to avoid any unjustified overhead, while preserving structural properties and proof preservation.

3 Code Patterns for local VM-Level Containers

Local VMLC are run-time entities that provide direct support for concurrent real-time behaviour based on the Ravenscar computation model. They are directly implemented by primitive services of the ASSERT VM, using the above described extension of the Ada Ravenscar profile. Four basic types of VMLC have been defined, based on the hard real-time terminal objects of HRT-HOOD [1]: cyclic, sporadic, protected, and passive containers.

VMLC are implemented using some meta-model elements, which are named after HRT-HOOD entities (figure 2):

- Provided and required interfaces,
- OPCS (operation control structure), embodying the functional behaviour of the component;
- OBCS (object control structure), providing synchronization mechanisms;
- Thread, providing the concurrent behaviour of the component.

The real-time and synchronization behaviour of a VMLC is provided by the OBCS and the thread. The OBCS is implemented by an Ada protected object, while the thread is implemented by an Ada task.

Code patterns that represent the common aspects of a class of model elements are called *archetypes*. Examples of archetypes are:

- Cyclic object, with or without deadline and overrun detection;
- Sporadic object, with or without deadline and overrun detection;

Appendix A includes some examples of such archetypes. More complex archetypes can built by combining these code patterns, e.g. a cyclic thread with both deadline miss and WCET overrun detection.

A key issue is how to handle temporal faults. Possible fault handling patterns include:

- *Error logging*. Although useful for testing or low integrity applications, this pattern is not suited for high-integrity tasks as it does not provide any kind of resilience to faults.
- *Second chance*. If there is some slack time available, the execution-time budget of the failing task can be extended for the current job. This is seldom acceptable in high-integrity systems, as it incurs some degree of temporal indeterminacy.
- *Mode change*. This is the most flexible approach in that it permits fault recovery to be performed as and when required.
- *Safe stop*. This strategy is often the only course to take for high-integrity systems.

A more extensive discussion of these schemes can be found in an upcoming paper [8].

4 Extending Patterns for Distribution

In this section we review Ada coding patterns for VMLC in the distributed case.

The support of distribution encompasses a variety issues that range from high-level middleware architecture down to implementation concerns. Some of those issues are al-ready discussed from an Ada perspective in PolyORB [13]; GLADE [7] and RT-GLADE projects. For the purposes of this paper we focus on interaction patterns, which are the most critical elements to warrant system analyzability.

We make the following hypotheses: a high-level model describes the deployment of the system; this view is used to build and dimension all tasks and buffers, in-place marshallers are built; a deterministic protocol is used [1]. Hence, we

[1] Please refer to the authors bibliography for more details on these hypotheses.

focus on how interaction patterns defined in the Ravenscar profile can be mapped onto interaction suitable for distribution.

We assume interactions are asynchronous and one-way, which is a sensible choice for Distributed Real-Time systems for it reduces global blocking time. Besides, this is compatible with typical requirements from critical systems and typical scheduling techniques such as [12] and its extensions. We assume that a one-to-one relation exists between the priority of the request (the message to be exchanged) and the priority of the processing tasks.

The Ravenscar profile prescribes that inter-task communication take place via protected objects. Extending this configuration to distribution can be as simple as "splitting" the protected object in two separate entities: one client-side activity that formats the request and passes it to the communication subsystem; one server-side activity that is notified a request is arriving. Depending on the enforcement policies of choice, a task can be released from a thread pool, or else a dedicated task is awakened. Let us briefly review each configuration in isolation (figure 3):

- **Client Side**: this configuration is similar to the local case: the client passes its request to the communication subsystem that sends this request to the remote node. Intermediate steps are required to marshall the payload, and to build proper messages. Depending on the protocol stack, an intermediate protected object might be placed between client code and the protocol-specific threads.
- **Server Side**: on the server side, the request is received by the communication subsystem and stored in some internal buffer. It is safe to assume that this buffer is embodied in a protected object (PO). Several scenarios are then possible:
  - *One PO per receiving task (1)*: this scenario is similar to the local case where at most one task is waiting per protected object;
  - *One PO plus a pool of tasks (2)*: this scenario implies that a task is waiting for the next request while some other tasks are either processing jobs or idling, or else blocked on a task-specific PO. This arrangement corresponds to the *Leader/Followers* or *Thread Pool* patterns [10]. In such a scenario it may be convenient to use different priorities for I/O and message sending as well as for request processing tasks, for example to discriminate between urgency and priority.

Furthermore, one must select patterns to dispatch to the correct subprogram, with two strategies *one PO per sub-program*, or *one PO shared by several subprograms* and skeleton-like code to dispatch to the correct subprogram. Each strategy has been implemented and made deterministic separately by using local Ravenscar patterns and a transport library. Those two elements can then be combined to form four different configurations. The selection of one particular configuration has to be made depending on two antonymous criteria: footprint size of the system vs. ease of analysis. At the time of this writing the ASSERT project team is still evaluating the costs and trade-offs of each configuration in term of pessimistic WCET and memory overhead.

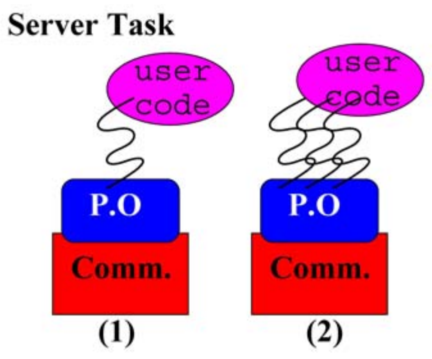


Figure 3 Request Handling Strategies

5 Conclusions and future work

In this paper we have briefly outlined the motivations and the strategy we have adopted for the definition of Ada 2005 coding patterns for the support of the code generation stage of an advanced model-driven development infrastructure. We are presently approaching the final implementation stage of our vision and we shall soon have performance figures useful to ascertain the viability of the approach we pursue.

References

[1] A. Burns and A. Wellings. *HRT-HOOD: A design method for hard-real-time*. Real-Time Sytsems, 6(1):73–114, 1994.

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

[3] J. A. de la Puente and J. Zamorano. *Execution-time clocks and Ravenscar kernels*. Ada Letters, XXIII(4): 82–86, December 2003. Proceedings of the 12th International Ada Real-Time Workshop (IRTAW12).

[4] J. A. de la Puente, J. Zamorano, J. F. Ruiz, R. Fernández, and R. García. *The design and implementation of the Open Ravenscar Kernel*. Ada Letters, XXI(1), 2001. Proceedings of the 10th International Real-Time Ada Workshop.

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

[6] ESA/ESTEC. ECSS-E-50-12A *SpaceWire - Links, nodes, routers and networks*. Technical report, European Space Agency, 2003.

[7] L. Pautet and S. Tardieu. *GLADE: a Framework for Building Large Object-Oriented Real-Time Distributed*

Systems. In Proceedings of the 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'00), Newport Beach, California, USA, June 2000. IEEE Computer Society Press.

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

[9] J. A. Pulido, S. Urueña, J. Zamorano, T. Vardanega, and J. A. de la Puente. *Hierarchical scheduling with Ada 2005*. In M. G. H. Lus Miguel Pinho, editor, Reliable Software Technologies - Ada-Europe 2006, volume 4006 of LNCS. Springer Berlin / Heidelberg, 2006. ISBN 3-540-34663-5.

[10] D. C. Schmidt and F. Buschmann. *Patterns, frameworks, and middleware: their synergistic relationships*. In ICSE '03: Proceedings of the 25th International Conference on Software Engineering, pages 694–704, Washington, DC, USA, 2003. IEEE Computer Society.

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

[12] K. Tindell. *Holistic schedulability analysis for distributed hard real-time systems*. Technical report, University of York, 1993.

[13] T. Vergnaud, J. Hugues, L. Pautet, and F. Kordon. *PolyORB: a schizophrenic middleware to build versatile reliable distributed applications*. In Proceedings of the 9th International Conference on Reliable Software Techologies Ada-Europe 2004 (RST'04), volume LNCS 3063, pages 106 – 119, Palma de Mallorca, Spain, June 2004. Springer Verlag.

A VM-level container archetypes

The code templates in the next pages illustrate the approach use in building component archetypes. More complex archetypes cab be built using this basic set.

Pattern 1. Cyclic thread archetype with deadline miss detection.

```
Deadline_Miss : Timing_Event ;
task type  Cyclic_Thread(P :  Priority; T : Time_Span ; D : Time_Span )  is
    pragma  Priority(P) ;
end  Cyclic_Thread;
task body  Cyclic_Thread  is
    Next_Time : Time : = Clock ;
    Deadline_Missed : Boolean ;
begin
    loop
        Deadline_Miss.Set_Handler ( Next _Time + D, Deadline_Miss _Handler ' Access ) ;
        delay until  Next _Time ;
        OPCS. Activity ;  --  periodic job
        Next_Time : = Next_Time + T ;
    end loop ;
end  Cyclic_Thread;
```

Pattern 2. Cyclic thread archetype with WCET overrun detection.

```
task type  Cyclic_Thread(P :  Priority; T : Time_Span ; D : Time_Span )  is
    pragma  Priority(P) ;
end  Cyclic_Thread;
task body  Cyclic_Thread  is
    Next_Time : Time : = Clock ;
    Id :  aliased constant  Task_Id : = Current_Task ;
    WCET_Timer : Ada.Execution_Time.Timers .Timer ( Id ' Access ) ;
begin
    loop
        Set_Handler ( WCET_Timer , C, Overrun_Handler ' Access ) ;
        delay until Next_Time ;
        OPCS. Activity ;  --  periodic job
        Next_Time : = Next_Time + T ;
    end loop ;
end  Cyclic_Thread;
```

Pattern 3. Sporadic thread archetype with minimal inter-arrival time enforcement.

```
OBCS : Sporadic_OBCS ( C e i l i n g ) ;
task type  Sporadic_Thread(P :  Priority; T : Time_Span)  is
    pragma  Priority(P) ;
end  Sporadic_Thread;
task body  Sporadic_Thread  is
    Next_Time : Time : = Clock ;
begin
    loop
        delay until Next_Time ;
        OBCS. Get_Request ;
        OPCS. Activity ;  --  sporadic job
        Next_Time : = Next_Time + T ;
    end loop ;
end  Cyclic_Thread;
procedure  Start  is
begin
    OBCS.Put_Request ;
end  Start ;
```

Ada Gems

The following contributions are taken from the AdaCore Gem of the Week series. The full collection of gems, discussion and related files, can be found at http://www.adacore.com/category/developers-center/gems/.

Ada Gem -5: Key-Based Searching In Set Containers

Matthew Heaney, On2 Technologies

Date: 11 June 2007

Abstract: Sets are containers of elements. Like all containers, they are searchable, and given an element value you can find the element in the set equivalent to that value. For some applications, element equivalence is determined by just a part of the element (its "key" part), and it is often necessary to find an element given only a key value. A technique is presented here for key-based searching of elements in set containers.

Let's get started…

This example demonstrates how to use hashed set containers.

We are given the task of implementing a game in which a user navigates among components in a maze. These map-sites comprise walls, rooms, and doors. One of the features of the game is the ability to find room objects given a room number, and we implement that feature using a set container.

The map-site objects in our simulation are members of a common class. We use a tagged type declared as follows:

```
type Map_Site is abstract tagged limited null record;
```

The (abstract) type has a single (abstract) operation to provide navigation to a site object:

```
procedure Enter (Site : in out Map_Site) is abstract;
```

Next we implement concrete type Room, which derives from Map_Site. The declaration of the Room type looks like this:

```
type Room (<>) is limited new Map_Site with private;
```

The room type is limited and indefinite (because it has "unknown discriminants"), which means that objects of the type must be explicitly initialized by calling a constructor-style function, declared as follows:

```
not overriding
function New_Room (Number : Positive) return not null
        access Room;
```

There is no way to create a room object other than to call New_Room, which is where we insert new room objects into the set used to find instances. Now we turn to the package body and the instantiation of the set package. The generic actual type is just a simple named access type (the same one we use to allocate instances):

```
type Room_Access is not null access Room;
```

To instantiate the (hashed) set package, we need both a hash function and an equivalence function. The hash function for sets is a mapping of element value to hash value. So how do we make a hash value from a room object? The natural choice is to use the room number as the hash value for a room object, so our hash function for rooms looks like this:

```
function Hash (R : Room_Access) return Hash_Type is
begin
   return Hash_Type (R.Number);
end;
```

For the equivalence function, we can just use the predefined equality operator for type Room_Access, since set elements are access values, and access values are unique for distinct room objects.

We now have everything we need to instantiate the generic hashed set container:

```
package Room_Sets is new
        Ada.Containers.Hashed_Sets
        (Element_Type          => Room_Access,
         Hash                  => Hash,
         Equivalent_Elements   => "=");

Room_Set : Room_Sets.Set;
```

We are finally in a position to implement the Room constructor function, which looks like this:

```
function New_Room (Number : Natural)
  return not null access Room
is
  R : constant Room_Access := new Room (Number);
begin
  Room_Set.Insert (R);
  return R;
end New_Room;
```

Now that we have implemented the necessary infrastructure for creating room objects, we have to implement the function for looking up room objects using the room number as the key. Our function is declared as follows:

```
function Find_Room (Number : Natural) return
        access Room;
```

Now comes the interesting part. The set records all Room objects that have been created, so if a room with that number exists, it will be in the set. Our problem is that we have no immediate way to search for room objects given a room number. Remember that this is a set of rooms (not a map with room number as the key) and the search function for the instantiation looks like this:

```
function Find (Container : Set; Item : Room_Access)
        return Cursor;
```

The issue is that we have a room number, not a room object, so how do we look up a room object according to its number?

The solution is to take advantange of the nested generic package Generic_Keys provided by the sets, which allows you to view a set element in terms of its key. It has generic formals very similar to the set package itself, except that the operations apply to a generic formal Key_Type instead of to the element type.

One requirement for using that package is that the generic actuals for keys must deliver the same values as the generic actuals for elements. For example, the function to return the hash value of a room number (the key) must return the same value as the function that returns the hash value of the room object (the element) with that room number. So we instantiate the nested package as follows:

```
function Get_Room_Number (R : Room_Access)
        return Natural is
begin
  return R.Number;
end;
function Room_Number_Hash (N : Natural)
        return Hash_Type is
begin
  return Hash_Type (N);
end;

package Room_Number_Keys is new
        Room_Sets.Generic_Keys
        (Key_Type          => Natural,
         Key               => Get_Room_Number,
         Hash              => Room_Number_Hash,
         Equivalent_Keys => "=");
```

We are now ready to implement the Find_Room function. The package Room_Number_Keys provides a key-based search function, so we call that to search the set for a room object with the given room number:

```
function Find_Room (Number : Natural)
        return access Room is
  C : constant Room_Sets.Cursor :=
      Room_Number_Keys.Find (Room_Set, Number);
begin
  if Has_Element (C) then
    return Element (C);
  else
    return null;
  end if;
end Find_Room;
```

The search function returns a cursor, and the cursor value indicates whether the search was successful. If a room object having that room number is in the set, Has_Element returns True and so we return the Room object designated by the cursor. If that number was not found (because no room object having that number was in the set), we simply return null.

The package Generic_Keys has a few other interesting features, including the ability to modify set elements, but we'll defer that discussion until a future time. In the meantime, have fun with the containers!

Ada Gem -8: Factory Functions

Matthew Heaney, On2 Technologies

Date: 10 September 2007

Abstract: A factory function is a technique for constructing an object from one class given only an object in some other class. The technique is used to implement assignment for objects having a class-wide type, such that tag-mismatch exceptions cannot occur.

Let's get started…

Suppose we have a generic package that declares a stack class. The root of the hierarchy would be as follows:

```
generic
  type Element_Type is private;
package Stacks is
  type Stack is abstract tagged null record;

  procedure Push (Container : in out Stack;
                  Item : in Element_Type) is abstract;
  …
end Stacks;
```

Assume there are various concrete types in the class, say an unbounded stack (that automatically grows as necessary) and a bounded stack (implemented as a fixed-size array).

Now suppose we want to assign one stack to another, irrespective of the specific stack type, something like this:

```
procedure Op (T : in out Stack'Class; S : Stack'Class) is
begin
  T := S;  -- raises exception if tags don't match
  …
end;
```

This compiles, but isn't very robust, since if the tag of the target stack doesn't match the tag of the source stack, then an exception will occur. Our goal here is to figure out how to assign stack objects (whose type is class-wide) in a manner such that the assignment is guaranteed to work without raising a tag-mismatch exception.

One way to do this is to make an assignment-style operation that is primitive for the type, so that it will dispatch according to the type of the target stack. If the type of the source stack is class-wide, then there can't be a tag mismatch (and hence no exception) since there's only one controlling parameter.

(Note that you could do it the other way too, by dispatching on the tag of the source stack. You could even make the operation class-wide, so that it doesn't need to dispatch at all. The idea is to avoid passing more than a single controlled operand.)

The assign operation would be declared like this:

```
procedure Assign (Target : in out Stack;
                  Source : Stack'Class) is abstract;
```

which would allow us to rewrite the above assignment statement as:

```
procedure Op (T : in out Stack'Class; S : Stack'Class) is
begin
  T.Assign (S);  -- dispatches according T's tag
  …
end;
```

Each type in the class will have to override Assign. As an example, let's follow the steps the necessary to implement the operation for the bounded stack type. Its spec would look like this:

```
generic
package Stacks.Bounded_G is
```

```
   type Stack (Capacity : Natural) is
      new Stacks.Stack with private;

   procedure Assign
    (Target : in out Stack;
     Source : Stacks.Stack'Class);
   …
private
   type Stack (Capacity : Natural) is
      new Stacks.Stack with
   record
     Elements  : Element_Array (1 .. Capacity);
     Top_Index : Natural := 0;
   end record;
end Stacks.Bounded_G;
```

This is just a canonical implementation of a bounded container form, that uses a discriminent to control how much storage for the object is allocated. The interesting part is implementing the Assign operation, since we need some way to iterate over items in the source stack. Here's a skeleton of the implementation:

```
procedure Assign
            (Target : in out Stack;  -- bounded form
             Source : Stacks.Stack'Class)
is
   …
begin
   …
   for I in reverse 1 .. Source.Length loop
     Target.Elements (I) := <get curr elem of source>
     <move to next elem of source>
   end loop;
   …
end Assign;
```

Note carefully that, assuming we visit items of the source stack in top-to-bottom order, it's not a simple matter of pushing items onto the target stack, since if we did that the items would end up in reverse order. That's the reason why we populate the target stack array in reverse, starting from largest index (the top of the stack) and working backwards (towards the bottom of the stack).

The question is, how do you iterate over the source stack? Assume that each specific type in the stack class has its own iterator type, matched to that stacks's particular representation (similar to how the containers in the standard library are implemented). The issue is that the type of the source stack formal parameter is class-wide. How do we get an iterator for the source stack actual parameter, if its specific type is not known (not known statically, that is)?

The answer is, just ask the stack for one! A tagged type has dispatching operations, some of which can be functions, so here we just need a dispatching function to return an iterator object. The idiom of dispatching on an object whose type is in one class, to return an object whose type is in another class, is called a "factory function" or "dispatching constructor."

An operation can only be primitive for one tagged type, so if the operation dispatches on the stack parameter then the function return type must be class-wide. We now introduce type Cursor, the root of the stack iterator hierarchy, and amend the stack class with a factory function for cursors:

```
type Cursor is abstract tagged null
   record;  -- the iterator

function Top_Cursor  -- the factory function
  (Container : not null access constant Stack)
  return Cursor'Class is abstract;

… -- primitive ops for the Cursor class
```

Each type in the stack class will override Top_Cursor, to return a cursor that can be used to visit the items in that stack object. We can now complete our implementation of the Assign operation for bounded stacks as follows:

```
procedure Assign (Target : in out Stack;
                  Source : Stacks.Stack'Class)
is
  C : Stacks.Cursor'Class :=
                   Source.Top_Cursor;  -- dispatches
begin
  Target.Clear;

  for I in reverse 1 .. Source.Length loop
    Target.Elements (I) := C.Element;  -- dispatches
    C.Next;  -- dispatches
  end loop;

  Target.Top_Index := Source.Length;
end Assign;
```

The Source parameter has a class-wide type, which means the call to Top_Cursor dispatches (since Top_Cursor is primitive for the type). This is exactly what we want, since different stack types will have different representations, and will therefore require different kinds of cursors. The cursor object (here, C) returned by the factory function is itself class-wide, which means that cursor operations also dispatch. The function call C.Element returns the element of Source at the current position of the cursor, and C.Next advances the cursor to the next position (towards the bottom of the stack).

Ada Gem -9: Classwide Operations, Iterators, and Generic Algorithms

Matthew Heaney, On2 Technologies

Date: 17 September 2007

Abstract: A generic algorithm manipulates container elements in a way that is completely general, working with any container (including arrays) that provides a way to iterate over its elements. In this example we systematically alter an operation for copying containers within a common class, converting it to a generic algorithm that ultimately works for any kind of container.

Let's get started…

In the last gem, we used a stack class to demonstrate factory functions (to construct iterator objects), and implemented an assignment operation that dispatched on the type of the target stack. We mentioned in passing that that operation could be implemented by dispatching on the source stack, so let's show how to do that.

We reorder the parameters so that the Source stack is first in the parameter list (so that it's the "distinguished receiver" of a prefix-style call), and change its type from classwide to specific. We also change the name of the operation from Assign to Copy, per convention. The new declaration is as follows:

```
procedure Copy (Source : Stack;
                Target : in out Stack'Class)

is abstract;
```

In the earlier example, we had to populate the target stack in reverse, so that the elements would be in the correct order. We were able to do that because the operation was implemented by the specific type, and hence it had direct access to the representation of the (target) stack. Here the target type is classwide, so the only way to populate it is in forward order, using Push. That means we'll have to iterate over the source stack in reverse, so that the items are properly ordered in the target.

The bounded stack type is implemented as an array, so implementing Copy is easy because the bottom of the stack begins at the beginning of the array:

```
procedure Copy
      (Source : Stack;  -- bounded stack (array-based)
       Target : in out Stacks.Stack'Class)
is
begin
  Target.Clear;

  for I in 1 .. Source.Top_Index loop    -- from bottom
                                         -- to top
    Target.Push (Source.Elements (I));  -- Elements is
                                         -- the array
  end loop;
end Copy;
```

We also said in the earlier gem that the operation need not be primitive for the type. If we change the source stack's type to classwide, then the operation itself becomes classwide:

```
procedure Copy2  -- classwide op, not primitive
  (Source : Stack'Class;
   Target : in out Stack'Class);
```

If we make the type of the source stack classwide, then we'll need a different way to iterate over items of the source stack in reverse order, since we don't have access to its representation anymore.

To do that we'll amend the cursor type to include some additional operations. First we'll add a new factory function, to construct a cursor object that (initially) designates the element at the bottom of the stack:

```
function Bottom_Cursor
        (Container : not null access constant Stack)
  return Cursor'Class is abstract;
```

We'll also need an operation to move the cursor to the element that precedes the current item:

```
procedure Previous (Position : in out Cursor)

is abstract;
```

That gives us everything we need to turn Copy into a classwide operation, so that it only needs to be implemented once:

```
procedure Copy2 (Source : Stack'Class;
                 Target : in out Stack'Class)
is
  C : Cursor'Class := Bottom_Cursor (Source'Access);
begin
  Target.Clear;

  while C.Has_Element loop
    Target.Push (C.Element);
    C.Previous;
  end loop;
end Copy2;
```

Note that we declared the classwide stack operation in the root package (see stacks.ads), but it could have just as easily been declared as a generic child procedure:

```
generic
procedure Stacks.Generic_Copy3
      (Source : Stack'Class;
       Target : in out Stack'Class);
```

Actually, we could move the operation out of the package hierarchy entirely:

```
with Stacks;
generic
  with package Stack_Types is new Stacks (<>);
  use Stack_Types;
procedure Generic_Stack_Copy4
 (Source : Stack'Class;
  Target : in out Stack'Class);
```

We can generalize this even more, such that the copy algorithm works for any kind of stack:

```
generic
  type Stack_Type (<>) is limited private;
  type Cursor_Type (<>) is private;
  type Element_Type (<>) is private;

  with function Bottom_Cursor  (Stack : Stack_Type)
     return Cursor_Type is <>;
  with procedure Push (Stack : in out Stack_Type;
                       Item  : Element_Type) is <>;
  with procedure Previous
       (Cursor : in out Cursor_Type)   is <>;
  …
procedure Generic_Stack_Copy5
       (Source : Stack_Type;
        Target : in out Stack_Type);
```

This illustrates the difference between the dynamic polymorphism of tagged types and the static polymorphism of generics. There is no need for a stack class anymore (having a dedicated copy operation that works only for types in that class), since the generic algorithm works for any stack. (This is exactly how the standard container library is designed. Container types are tagged, but they are not members of a common class.)

Instantiating this operation on our stack type is easy, since the names of the generic actual operations match the names of the generic formal operations, so we don't need to specify them explicitly (since the generic formals are marked as accepting a <> default):

```
procedure Test_Copy5 (S : Stack) is
  procedure Copy5 is
    new Generic_Stack_Copy5
              (Stack,
               Cursor,
                Integer); -- default everything else

  T : Stack (S.Length);
begin
  Copy5 (Source => S, Target => T);
end;
```

But why stop there? We can write a generic copy algorithm for any kind of container. We just need to generalize iteration a little, to mean "visit these items in the way that makes sense for this source container," and generalizing insertion, to mean "add this element in the way that makes sense for this target container." The declaration would be:

```
generic
  type Container_Type (<>) is limited private;
  type Cursor_Type (<>) is private;
  type Element_Type (<>) is private;

  with function First (Container : Container_Type)
     return Cursor_Type is <>;
  with procedure Insert
      (Container : in out Container_Type;
       Item      : Element_Type) is <>;
  with procedure Advance (Cursor : in out Cursor_Type)
   is <>;

procedure Generic_Copy6
        (Source : Container_Type;
         Target : in out Container_Type);
```

We can instantiate this using our stack type, but note that the generic actuals no longer match the generic formals, so we need to specify them explicitly:

```
procedure Test_Copy6 (S : Stack) is
  procedure Copy6 is
    new Generic_Copy6 (Stack, Cursor, Integer,
                       First     => Bottom_Cursor,
                       Insert    => Push,
                       Advance   => Previous);

  T : Stack (S.Length);

begin
  Copy6 (Source => S, Target => T);
end;
```

One assumption we've made here is that the source and target containers have the same type. Suppose we would like to copy the items in a stack to, say, an array. One approach would be to introduce another generic formal container type (a "source container" type that is distinct from the "target container" type), but there's another way. Consider the implementation of the copy algorithm:

```
procedure Generic_Copy6
      (Source : Container_Type;
       Target : in out Container_Type)
is
  C : Cursor_Type := First (Source);
begin
  Clear (Target);
  while Has_Element (C) loop
    Insert (Target, Element (C));
    Advance (C);
  end loop;
end Generic_Copy6;
```

Notice that the only thing we do with the source container is to use it to construct a cursor. If we pass in the cursor directly, that eliminates any mention of the source stack, which in turn allows the source and target containers to be different types. Our algorithm now becomes:

```
generic
  type Container_Type (<>) is limited private;
  type Cursor_Type (<>) is private;
  type Element_Type (<>) is private;
  …
procedure Generic_Copy7
      (Source : Cursor_Type;
       Target : in out Container_Type);
```

We can now copy from an integer stack to an array like this:

```
procedure Copy_From_Stack_To_Array
     (S : in out Stack)
is
  T : Integer_Array (1 .. S.Length);
  I : Positive := T'First;

  procedure Insert
       (Container : in out Integer_Array;
        Item      : Integer)
  is
  begin
    Container (I) := Item;
    I := I + 1;
  end;

  procedure Copy7 is
    new Generic_Copy7   (Integer_Array,  Cursor,
                         Integer,
                         Advance => Next);
begin
  Copy7 (Source => S.Top_Cursor, Target => T);
end Copy_From_Stack_To_Array;
```

The target "container" is just an array. The only special thing we need to do is synthesize an insertion operation, to pass as the generic actual. We can also use the same algorithm to go the other way, from an array to a stack:

```
procedure Copy_From_Array_To_Stack
     (S: Integer_Array)
is
  T : Stack (S'Length);

  function Has_Element (I : Natural) return Boolean is
```

```
   begin
     return I > 0;
   end;

   function Element (I : Natural) return Integer is
   begin
     return S (I);
   end;

   procedure Advance (I : in out Natural) is
   begin
     I := I - 1;
   end;

   procedure Copy7 is
      new Generic_Copy7 (Stack, Natural,  Integer,
                         Insert => Push);

 begin
   Copy7 (Source => S'Last, Target => T);
 end Copy_From_Array_To_Stack;
```

Now the source container is an array, and the "cursor" is just the array index (an integer subtype). We have the familiar problem of ensuring that the target stack is populated in the correct order. As before, we simply iterate over the array in reverse, by passing the index S'Last as the initial cursor value, and then "advancing" the cursor by decrementing the index value.

The algorithm can be generalized further still. In this final version, we eliminate the generic formal element type. That means we'll need to modify the generic formal Insert operation, by passing the source cursor as a parameter instead of the source element. The declaration of the generic algorithm now becomes:

```
 generic
   type Container_Type (<>) is limited private;
   type Cursor_Type (<>) is private;

   with procedure Insert
        (Target : in out Container_Type;
         Source : Cursor_Type) is <>;
   …
 procedure Generic_Copy8
        (Source : Cursor_Type;
         Target : in out Container_Type);
```

The algorithm is now agnostic about the mapping from cursor to element (since it doesn't even know about elements), which is more flexible, since it allows the client to choose whatever mechanism is the most efficient. To use the new algorithm, all we need to do is make a slight change to the generic actual Insert procedure, as follows:

```
 procedure Copy_From_Stack_To_Array
      (S : in out Stack)
 is
   T : Integer_Array (1 .. S.Length);
   I : Positive := T'First;

   procedure Insert
        (Target : in out Integer_Array;
         Source : Cursor)  -- now a cursor instead of
                           -- an element
   is
   begin
     Target (I) := Element (Source);
     I := I + 1;
   end;

   procedure Copy8 is
     new Generic_Copy8 (Integer_Array, Cursor,
                        Advance => Next);
 begin
   Copy8 (Source => S.Top_Cursor, Target => T);
 end Copy_From_Stack_To_Array;
```

The basic idea is that a generic algorithm can be used over a wide range of containers (including array types). A cursor provides access to the elements in a container, but as we've seen, once you have a cursor then the container itself sort of disappears. From the point of view of a generic algorithm, a container is merely a sequence of items.

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o K.U. Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Peter Dencker
Steinäckerstr. 25
D-76275 Ettlingen-Spessartt
Germany
Email: dencker@web.de
URL: ada-deutschland.de

Ada-France

Association Ada-France
c/o Jérôme Hugues
Département Informatique et Réseau
École Nationale Supérieure des Télécomunications
46, rue Barrault
75634 Paris Cedex 135
France
Email: bureau@ada-france.org
URL: www.ada-france.org

Ada-Spain

attn. José Javier Gutiérrez
Ada-Spain
P.O.Box 50.403
28080-Madrid
Spain
Phone: +34-942-201-394
Fax: +34-942-201-402
Email: gutierjj@unican.es
URL: www.adaspain.org

Ada in Sweden

attn. Rei Stråhle
Saab Systems
S:t Olofsgatan 9A
SE-753 21 Uppsala
Sweden
Phone: +46 73 437 7124
Fax: +46 85 808 7260
Email: Rei.Strahle@saabgroup.com
URL: www.ada-i-sverige.se

Ada in Switzerland

attn. Ahlan Marriott
White Elephant GmbH
Postfach 327
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: ada@white-elephant.ch
URL: www.ada-switzerland.ch