ADA USER JOURNAL

Volume 28
Number 4
December 2007

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

Further details on our approach to these are given below.

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

News and Product Announcements

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

Reprinted Articles

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

Editorial

This issue closes volume 28 of the Ada User Journal; at the same time as the year 2007 comes to an end. Not looking back, but forward, for sure the highlight of 2007 was the approval by ISO of the (officially known as) Amendment to ISO/IEC 8652, informally and better known as Ada 2005. Not being a change as extensive as it was Ada 95, there are considerable improvements to the language, particularly in the Object Oriented and Real-Time Systems domains, allowing us to look forward to a successful 2008. This approval was the conclusion of a long process that had the contribution of the large majority of the Ada community. We should acknowledge this effort, but in particular thank the members and editor of the Ada Rapporteur Group, the main responsibles for "pushing" the process forward. The approval was also the start of a new process; do not forget that Ada will be what we make it. It is now the job of each one of us to advertise its merits and encourage its use.

As for this last issue of 2007, I am sure that you will find its contents worthwhile. The first paper is a contribution from the Industrial Track of the Ada-Europe 2007 conference, by Igor Furgel and Lars Hanke of T-Systems, Germany, describing a business process for managing secure updates in embedded systems.

Next, and taking the main share of the issue, is the first part of the Proceedings of the 13th International Real-Time Ada Workshop (IRTAW-13) that, as promised, will be reprinted in the Journal. This issue provides the contents of the first session of the workshop: Language Issues. The first paper is the session report, which describes the main debate and conclusions of the session. Afterwards, you can find two papers coming from the University of York, UK. The first paper proposes a correction to the EDF protocol specification in Ada 2005, while the second proposes to augment Ada 2005 by allowing requeuing via an interface. Finally, the last paper of the session, coming from the Technical University of Madrid, Spain, discusses restrictions and additions in order to support high-integrity hard real-time distributed applications.

In this issue we also start publishing contributions from the Gem of the Week series, with the permission of AdaCore, for which I am thankful. Inaugurating the section is the series of gems concerning Limited Types in Ada 2005, by Bob Duff. Finally, the News, Calendar and Forthcoming Events sections complete the issue.

Before concluding; some of you may have noticed that there were problems with the printing process of the Journal, which led to quality problems in some of the copies of the September issue. Unfortunately we did not detect this before shipment, for which I apologise. For the December issue we introduced an extra verification step in the production process that we hope will prevent similar problems in the future. In the meanwhile, if you deem necessary please contact the Deputy Editor of the Ada User Journal, Jorge Real, as we are inquiring the printer on the possibility of printing extra copies.

Concluding this last editorial of 2007, I wish all the Journal collaborators and readers the best for 2008!

Luís Miguel Pinho
Porto
December 2007
Email: lmp@isep.ipp.pt

News

Santiago Urueña

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

Contents

Ada-related Events

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

Oct 2 — Ada day at Saab Sweden

From: AdaCore
Date: September 28, 2007
Subjet: Ada day at Saab Sweden
URL: http://www.adacore.com/2007/09/28/ada-day-at-saab-sweden/

Ada day at Saab Sweden

October 2, Järfälla, Sweden

Agenda Tuesday October 2:

09.30 – 10.30 Quality Assurance with GNAT Pro

A review of the functionalities of the GNAT Pro toolsuite that help insure the quality of generated code. Emphasis will be put on the creation of coding style rules, the integration of unit testing and coverage in the development cycle, and the generation of metrics and browsable documentation.

10.30 – 10.45 Break

10.45 – 11.15 Help for Certification with the GNAT Pro High-Integrity Edition

A presentation of the various functionalities that can be of interest when developing software requiring certification: Coding standard definition and verification, static worst case analysis (stack usage and timing), robustness and consistency testing, source to object traceability, and runtime footprint and deactivated code reduction, etc.

11.15 – 11.45 The Double Life of Ada

Created by the DoD, Ada has a worldwide reference list of being used in large, embedded aerospace & defence projects. Less well known is the fact that Ada has a strong presence outside of these "traditional" markets.

11.45 – 12.00 Questions and answers

12.00 – 13.00 LUNCH

13.00 – 14.00 The GNAT Pro Foundry

An insight into the software factory that produces two GNAT Pro releases each year and a GNAT Pro wavefront every day on more than 30 different configurations. The life of an AdaCore support ticket from its creation in GNAT Tracker to its complete resolution, and its participation in the quality of the product.

14.00 – 14.15 Break

14.15 – 14.45 The GNAT Pro Roadmap

New features, new tools, new libraries, new supported configurations, new services…

14.45 – 15.00 Questions and answers

Nov 7 — SIGAda Awards

From: John McCormick <mccormick@cs.uni.edu>
Date: Thu, 20 Sep 2007 14:09:43 −0700
Subject: Call for SIGAda Award Nominations
Newsgroups: comp.lang.ada

Dear Members of the Ada Community:

On Wednesday, 7 November 2007, the 2007 SIGAda Awards will be presented in a special morning plenary session at the SIGAda 2007 conference in Fairfax, VA. (See http://www.acm.org/sigada/conf/sigada2007/ if you have somehow missed announcements of this year's annual SIGAda international conference.)

We welcome your nominations of deserving recipients.

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

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

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

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

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

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

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

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

Call our attention to the people who are most deserving, by nominating them. And please nominate by OCTOBER 15!

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

John McCormick

Chair ACM SIGAda

[See also "15 November — SIGAda Awards" in AUJ 27.4 (Dec 2006), p.197 —su]

Nov 4–9 — SIGAda 2007 Conference

From: John McCormick <mccormick@cs.uni.edu>
Date: Fri, 19 Oct 2007 13:04:49 −0700
Subject: SIGAda 2007 extended early registration rates

Newsgroups: comp.lang.ada

SIGAda 2007 has extended its early registration rates through November 1st! This allows everyone to take advantage of the lower rates during the time that the government is working under a Continuing Resolution (CR) for appropriations.

Note that current policy regarding a Continuing Resolution does not prohibit attendance at conference tutorials, workshops, and sessions which are directly related and relevant to current duties and responsibilities of software engineers, information technology specialists, IT security engineers, managers, etc. Such attendance is justified and can currently be funded by the federal government. The SIGAda 2007 conference falls under this policy.

As the SIGAda 2007 conference is scheduled shortly after the government's new fiscal year, we encourage all federal government employees and contractors to take this opportunity to attend and register using the early registration rates, now extended through November 1st to everyone. More information is available on our website at http://www.sigada.org/conf/sigada2007/

SIGAda 2007 includes Tutorials on Exposing Ada Web Services Using a Service-Oriented Architecture (SOA), Ada 2005, Security by Construction, and Languages for Safety-Critical Software: Issues and Assessment DO178/DO278. It also includes Technical Papers such as High Assurance Profile for CORBA, Neural Networks, and SPARK Ada. Exhibitors are Lockheed Martin, AdaCore, OC Systems, Praxis, Northrop Grumman, Telelogic, Genco Systems, ARTiSAN Software, Ellidiss Software, Aonix, and Integrated Computer Solutions, Inc.

The workshops include a Two-Day Summit on Software Analysis sponsored by NIST and the Department of Homeland Security. All the Workshops are free with a minimum one-day conference registration. The conference will have a grand Banquet on Tuesday evening with outstanding Ethnic Cultural Performances.

As a recent note of interest to the Ada community, the Federal Aviation Administration (FAA) has just announced Government Acceptance of a multi billion dollar Ada based core Air Traffic Control System developed on budget and ahead of schedule (see http://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=7714 for details).

[See also "12–16 November — SIGAda 2006 Conference" in AUJ 27.4 (Dec 2006), pp.197–198 —su]

Dec 6 — Ada-France 2007

From: Frank Singhoff <singhoff@univ-brest.fr>
Date: Wed, 14 Nov 2007 00:39:03 +0100
Subject: Programme de la journée Ada-France 2007: méthodes, processus, modèles et outils pour l'ingénierie du logiciel embarqué temps réel critique
Newsgroups: fr.comp.lang.ada

[Translated from French. —su]

Program of the Ada-France technical Seminar 2007: methods, processes, models and tools for engineering embedded critical real-time software

Theme of the Seminar

An embedded system is a coherent set of components (hardware and software). It is often invisible to the equipment's user. It provides to the equipment the ability to complete a set of specific tasks by providing it intelligence. It performs data processing and manages the equipment's exchange of information necessary to accomplish its tasks. On the other hand, an embedded computer system faces, in varying degrees, to real-time constraints distinguishing the hard or critical real-time systems (need to meet response time deadlines, failing which the mission of the equipment can not be fulfilled) and soft real-time systems (trying to respect the constraints where non-compliance causes temporary dysfunction that does not cause the failure of the equipment's mission). We talk about critical embedded systems when the equipment performs a mission whose failure has the potential for a major impact on the lives, health of people, on the environment ... more generally on the fulfillment of the critical missions in which it participates. Whether because of their cost of production, their criticality, their complexity or their inaccessibility during use, the engineering of such systems requires the use of specific methods, processes, models and tools.

Seminar Program

Morning

9 h-9h30. Participants greeting, coffee.

9 h30-10h15. . Premiers retours d'un chercheur sur l'utilisation de l'IDM pour le temps réel. Jerome Delatour ESEO (Angers)

10 h15-11h. AADL: état et perspectives. Pierre Dissaux, Ellidiss Technologies (Brest)

11 h-11h45. Validation de systèmes temps-réel et embarqué à partir d'un modele MARTE: expérimentation. Eric Maes, Thales Research and Technology (Palaiseau)

11 h45-12h30. AUTOSAR: Streamlining automotive systems and processes. Francois Dupont, Geensys (Brest)

Afternoon

Lunch, coffee.

14 h30-15h15. Expérimentation d'unités de preuve pour la validation formelle de logiciels embarqués critiques. * Philippe Dhaussy, Pierre Yves Pilain * * Stephane Kerjean Dominique de Belloy **, Arnaud du Sorbier Monégier **, Hugues + Bonnin, Frederic Boniol ***. * Laboratory DTN, ENSIETA (Brest), ** Thales AIR SYSTEMS (Rungis), + CS-SI (Toulouse), *** IRIT-ENSEEIHT (Toulouse).

15 h15-16h. Ada 2005 pour les systèmes embarqués temps réel. Jose F. Ruiz, AdaCore (Paris)

16 h-16h45. Les outils de retro-ingénierie de code Ada. Eric Audrezet, Sodius (Nantes).

16 h45. End of the seminar, balance and announce of the next seminar.

Registration and information practices

Participation is free. The coffee breaks and a lunch will be offered by the sponsors. For reasons of logistics (meals, rooms), registration is mandatory. The application deadline is December 1st.

Registration can be made by returning the form below via email to F. Singhoff (singhoff@univ-brest.fr). [...]

The [Ada-France] seminar is held at the ENST. A map (plane, bus, car, train) is available here:

Http://www.enst-bretagne.fr/ecole/Campus_de_brest/ plan_d_acces/

Sponsors and Partners:

Ada-Core (http://www.adacore.com)

Ellidiss technologies (http://www.ellidiss.com)

Jessica-France (http://www.jessica-france.fr/)

Organizing Committee: (http://www.ada-france.org)

J. Hughes (President of Ada-France, ENST Paris, hugues@enst.fr)

Y. Kermarrec (Secretary of Ada-France, Enst-Bretagne, yvon.kermarrec@enst-bretagne.fr)

F. Singhoff (Treasurer of Ada-France, LISYC / University of Brest, singhoff@univ-brest.fr)

Jun 16–20 — Ada-Europe 2008

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Newsgroups: comp.lang.ada, fr.comp.lang.ada,comp.lang.misc
Subject: Ada-Europe 2008 submission deadline approaching
Date: Thu, 25 Oct 2007 22:03:36 +0200 (CEST)

Organization: Ada-Europe, c/o Dept. of Computer Science, K.U.Leuven
Summary: 17 days until submission deadline!
Keywords: Conference,tutorials,reliable software,Ada,LNCS,Venice,Italy

2nd CALL FOR PAPERS

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

Organized by Ada-Europe, in cooperation with ACM SIGAda (approval pending)

*** DEADLINE 11 NOVEMBER ***

Ada-Europe organizes annual international conferences since the early 80's. This is the 13th event in the Reliable Software Technologies series, previous ones being held at Montreux, Switzerland ('96), London, UK ('97), Uppsala, Sweden ('98), Santander, Spain ('99), Potsdam, Germany ('00), Leuven, Belgium ('01), Vienna, Austria ('02), Toulouse, France ('03), Palma de Mallorca, Spain ('04), York, UK ('05), Porto, Portugal ('06), Geneva, Switzerland ('07).

General Information

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.

Schedule

- 11 November 2007: Submission of regular papers, tutorial and workshop proposals
- 13 January 2008: Submission of industrial presentation proposals
- 03 February 2008: Notification to all authors
- 02 March 2008: Camera-ready version of regular papers required
- 11 May 2008: Industrial presentations, tutorial and workshop material required
- 16-20 June 2008: Conference

Topics

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

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

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

Call for Regular Papers

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

Proceedings

The authors of accepted regular papers shall prepare camera-ready submissions in full conformance with the LNCS style, not exceeding 14 pages and strictly by *2 March 2008*. For format and style guidelines authors should refer to: http://www.springer.de/comp/lncs/authors.html. Failure to comply and to register for the conference will prevent the paper from appearing in the proceedings. The conference proceedings will be published in the Lecture Notes in Computer Science (LNCS) series by Springer Verlag and will be available at the start of the conference.

Awards

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

Call for Industrial Presentations

The conference also seeks industrial presentations which may 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 by 13 January 2008. The Industrial Program Committee (yet to be named) 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.

Call for Tutorials

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

Call for Workshops

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

Call for Exhibitions

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

Discounts for Students

A limited number of grants are available for students who will co-author papers accepted at the conference. The grant will entail a reduction of 25% in the conference fee. Contact the Conference Chair for details.

Organizing Committee

Conference Chair

Tullio Vardanega, Università di Padova, Italy (tullio.vardanega@math.unipd.it)

Program Co-Chairs

Tullio Vardanega, Università di Padova, Italy (tullio.vardanega@math.unipd.it)

Fabrice Kordon, Université P. & M. Curie, France (fabrice.kordon@lib6.fr)

Tutorial Chair

Jorge Real, Universidad Politécnica de Valencia, Spain (jorge@disca.upv.es)

Exhibition Chair

Ahlan Marriott, White-Elephant GmbH, Switzerland (ada@white-elephant.ch)

Publicity Chair

Dirk Craeynest, Aubay Belgium & K.U.Leuven, Belgium (dirk.craeynest@cs.kuleuven.be)

Local Chair

Sabrina De Poli, Sistema Congressi srl, Italy (ae08@sistemacongressi.com)

Objektum — Embedded Systems Show

From: Objektum News & Events
Date: October 17th, 2007
Subject: Technology and Toolset Convergence Promoted at ESS 2007
URL: http://www.objektum.com/objektum/indexnews.asp?NewsID=477

Objektum (www.objektum.com), a leading provider of tailored training, consulting and software development for the aerospace and defence sector is promoting its vision for Application Lifecycle Management (ALM) toolset and technology convergence at the Embedded Systems Show (ESS) 2007.

Using its unique position in the discipline of software engineering and its UML / SysML toolset expertise, Objektum is working to integrate tools and technology to maximise the efficiency of managers, analysts, developers and testers. One such example is the development of two plug-ins for the acclaimed ARTiSAN Studio® toolset.

AdaCore — Systems & Software Technology Conference

From: AdaCore Press Center
Date: Wednesday October 24, 2007
Subject: Systems & Software Technology Conference (SSTC 2008)
RSS: http://www.adacore.com/2007/10/24/systems-software-technology-conference-sstc-2008/

Systems & Software Technology Conference (SSTC 2008)

AdaCore will be exhibiting at this event.

AdaCore — Embedded Systems Conference Silicon Valley

From: AdaCore Press Center
Date: Wednesday October 24, 2007
Subject: Embedded Systems Conference Silicon Valley
RSS: http://www.adacore.com/2007/10/24/embedded-systems-conference-silicon-valley-2/

Embedded Systems Conference Silicon Valley

AdaCore will be exhibiting at this event.

Ada Semantic Interface Specification (ASIS)

ASIS for Debuggers?

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Fri, 26 Oct 2007 12:09:27 +0200
Subject: Re: ASIS?
Newsgroups: comp.lang.ada

> If one were to need an API to fulfill all requirements by a Debugger, Source Browser and a Syntax-aware editor from the IDE, then would ASIS be it?

The debugger does not need all the capabilities of ASIS; only line numbers and type descriptions. That's what GDB gets from the object files generated by GNAT with -g. The editor does not need all the capabilities of ASIS. For example, emacs, GPS and Eclipse all do syntax highlighting without ASIS.

The source browser is probably the one component that would benefit from ASIS the most; however an alternative is gnatfind which uses the .ali files generated by GNAT for cross-references. GPS uses gnatfind, not ASIS, for this functionality. In contrast, adabrowse generates an HTML description of a program, with hyperlinks, using ASIS.

However, the answer to your question is probably yes: ASIS would provide all the information needed, and more, to the debugger, browser and editor, in a single interface.

> Can a compilation environment just provide an ASIS interface for third-party debuggers, source browsers and syntax-aware editors to plug-in and work well?

Yes but the word "just" is often the sign of a mistake and rings an alarm bell in my head whenever I see or hear it :) In this case, it hides these problems:

1) The complexity of providing the ASIS interface and that of using it from all tools.

2) The ASIS interface can only be provided on legal, compiling program text. Any program with compile-time errors in it would be impossible to browse (using the source browser) and the editor would have to be particularly smart in deciding when to call the compiler to regenerate the ASIS information.

3) In the worst of cases, regenerating the ASIS data can cause massive recompilations and be too slow for interactive use.

> Is a C interface available?

Not that I know. [...]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Fri, 26 Oct 2007 19:15:06 −0500
Subject: Re: ASIS?
Newsgroups: comp.lang.ada

> I thought ASIS only has "high-level" information prior to the target-dependent object-code generation and linking. I think the debugger will need the link map, information about the stack frame layouts, record layouts, etc, which is not in the ASIS domain. Am I wrong?

No. The Rationale for ASIS appendix of the ASIS standard states "It is not a goal to support tools having dynamic run-time requirements (e.g., symbolic debugger)." It can't be any clearer than that.

(Sorry I can't give you a reference to the standard so you can see for yourself, as the ASIS standard is copyright ISO and thus cannot be made publicly available.)

Ada and Education

Free book on multitasking

From: Bo Sanden <bsanden@acm.org>
Date: Sat, 29 Sep 2007 08:52:12 −0700
Subject: Free book on multitasking
Newsgroups: comp.lang.ada

Title: MULTITHREADING

Version: 0.9 August 2007

Author: Bo Sanden

License: Permission to copy if author, title and version are acknowledged

Copyright 2007 Bo Sanden, Colorado Technical University

URL: http://home.earthlink.net/~bosanden/Multithreading

This book is intended for designers and programmers of multitask software. It introduces the tasking/threading support in Ada and Java and presents Entity-Life Modeling, which is an intuitive design approach for reactive systems.

Reactive systems include those that operate in real time in the widest sense, such as embedded control systems as well as telephone switches; interactive systems from automated teller machines and gas pumps to travel reservation systems; and event-processing systems such as many games. ELM also applies to discrete-event simulations based on the process interaction worldview. ELM finds multitask solutions to problems that are inherently concurrent.

With Entity-life modeling, you pattern tasks on event threads in the problem domain much as an object-oriented program is patterned on objects in the problem domain. Most examples are in Ada.

Part I: Foundations

1. Introduction
2. Support for multithreading (Ada 2005, Java and Pthreads)
3. State modeling (practical use of state machines)

Part II: Entity-life modeling

4. Entity-life modeling
5. State-machine implementations
6. Resource sharing
7. Simultaneous exclusive access to multiple resources

Part III: Background and discussion

8. Real-time software architectures. Data-flow design approaches
9. The origins of entity-life modeling

Ada Intern Program

From: AdaCore Developer Center
Date: Tuesday September 18, 2007
Subject: Ada Intern Program
RSS: http://www.adacore.com/2007/09/18/ada-intern-program/

The new look Ada Intern Program is now up and running, providing a unique framework for AdaCore customers to source Ada-knowledgeable students from around the world for internship positions.

Customers can post internship offers and view applicants' profiles via their GNAT Tracker account.

Click here (http://www.adacore.com/home/academia/intern) for more information.

[See also "AdaCore Partners with Praxis Critical Systems on a Joint Academic Initiative" in AUJ 25-4 (Dec 2004), p.179. —su]

Webminar: GNATbench

From: AdaCore Press Center
Date: Thursday October 25, 2007
Subject: Webinar: GNATbench — The GNAT Pro Ada plug-in for Wind River Workbench
RSS: http://www.adacore.com/2007/10/25/online-event-webinar-gnatbench-the-gnat-pro-ada-plug-in-for-wind-river-workbench/

AdaCore has recently launched GNATbench 2.0, a significant upgrade with new capabilities in support of Wind River Systems' Workbench. The upgrade provides development teams using Workbench with advanced Ada language support and a fully integrated GNAT Pro Ada toolset to facilitate multi-language development, sophisticated editing, browsing, debugging, and comprehensive compilation. All versions of Ada are included — Ada 83, Ada 95, and Ada 2005 — for both kernel-module and real-time process (RTP) applications. Of special interest is the Code Assist editor feature that proposes identifier completions after a dot is entered, and formal parameter completions after an opening parenthesis is entered.

This webinar will appeal to Ada developers that are using, or are interested in using, GNAT Pro and the Wind River Workbench development environment in their projects. For more information on GNATbench please visit the GNATbench product page or contact sales@adacore.com.

From: AdaCore Developer Center
Date: Saturday November 10, 2007
Subject: GNATbench for Workbench Webinar
RSS: http://www.adacore.com/2007/11/10/gnatbench-for-workbench-webinar/

The archived training webinar featuring GNATbench for Workbench is now available for download. Please [go to http://www.adacore.com/home/gnatpro/webinars] to access it.

Webminar: Eclipse

From: AdaCore Press Center
Date: Thursday October 25, 2007
Subject: E-cast: Eclipse: Open standards
RSS: http://www.adacore.com/2007/10/25/e-cast-eclipse-open-standards/

COTS products come together to streamline safety-critical and OEM embedded development.

Summary

Eclipse has promised tools can plug in and work together, and we'll see concrete examples in this live event. Telelogic, AdaCore, and LynuxWorks are utilizing the power of the open Eclipse framework to help integrate and streamline the process from design to deployment.

Hear Telelogic explain how Rhapsody, a UML design tool, and DOORS, the industry standard requirements tool, are taking advantage of Eclipse to provide an easy path from design to code. AdaCore introduces their Eclipse-based GNATbench and GNAT Pro tool set that brings both Ada and C/C++ code together, integrating with the leading safety critical open-standards based RTOS — LynxOS. LynuxWorks shows how their Luminosity tool suite continues the common look and feel of Eclipse, to help build, test and deploy safety critical systems.

Webinar: GNAT Pro for OpenVMS

From: AdaCore Developer Center
Date: Thursday November 8, 2007
Subject: Webinar: GNAT Pro for OpenVMS on HP Integrity servers
RSS: http://www.adacore.com/2007/11/08/webinar-gnat-pro-for-openvms-on-hp-integrity-servers-2/

GNAT Pro for OpenVMS on HP Integrity servers training webinar — Nov 20, 2007.

AdaCore's GNAT Pro for HP OpenVMS I64 is a full-featured Ada development environment offering a natural migration path for Ada applications from other platforms. It includes a compiler that can handle all three versions of the Ada standard — Ada 83, Ada 95, and Ada 2005 — and provides a rich set of auxiliary tools and an extensive set of libraries. Ada-aware debugging is provided through HP's OpenDebug.

In this half-hour webinar AdaCore, with HP's participation, will describe the features and benefits of the GNAT Pro Ada development environment on HP OpenVMS I64 and answer questions from the audience. A particular focus will be on how to port Ada code from HP Ada on VAX and Alpha servers to GNAT Pro on OpenVMS I64. GNAT Pro supplies an extensive set of pragmas and attributes that are compatible with HP Ada, and for most Ada code the porting process should

be reasonably straightforward. For those situations where the program makes architectural assumptions that do not apply to I64 (for example 32- bit addresses) the webinar will identify the issues and offer effective solutions.

Webinar: GNAT Programming Studio InSight

From: AdaCore Developer Center
Date: Tuesday November 20, 2007
Subject: GNAT Programming Studio InSight webinar
RSS: http://www.adacore.com/2007/11/20/gnat-programming-studio-insight-webinar/

GNAT Programming Studio InSight webinar

Tuesday, December 11, 2007

9:00 am Pacific Standard Time (GMT −08:00, San Francisco)

12:00 pm Eastern Standard Time (GMT −05:00, New York)

6:00 pm Europe Standard Time (GMT +01:00, Paris)

In this latest webinar in the GNAT Pro InSight series, we will be demonstrating several new features present in recent versions of GPS using our latest release, GPS 4.2.0. Among the long list of new features and improvements that we will focus on are Remote Programming, Automatic Source Code Completion, Code Coverage support using gcov, an improved documentation generator, support for refactoring Ada sources, and an improved source code editor with enhanced source navigation and analysis capability. To enroll, please [go to http://adacore.webex.com/... —su]

Ada-related Tools

Artificial Intelligence libraries

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Thu, 04 Oct 2007 22:02:43 +0200
Subject: Re: Librairies d'IA
Newsgroups: fr.comp.lang.ada

[Translated from French. —su]

> I am in charge of a course of Artificial Intelligence for engineering students. I want them to program some heuristic or other in tutorial classes, and the language they learn in parallel to the course is Ada.

> Are Ada libraries in connection with the AI service for free?

> I am thinking in particular of libraries:
> - Genetic algorithms or equivalent (ant colonies, ...),
> - Neural Networks,
> - Programming by constraints,
> - Planning (A*)

http://sourceforge.net/projects/nocr

[Neuronal Optical Character Recognition: It's a tool who shows the concepts of a type of neuronal networks (multi-layers percetron). —su]

From: Jeffrey R. Carter <jrcarter@acm.org>
Newsgroups: fr.comp.lang.ada
Subject: Re: Librairies d'IA
Date: Fri, 05 Oct 2007 05:07:14 GMT

> - Genetic algorithms

PragmARC.Genetic_Algorithm

> - Neural Networks,

PragmARC.REM_NN_Wrapper

http://pragmada.home.mchsi.com/

From: Jean-Pierre Rosen <rosen@adalog.fr>
Newsgroups: fr.comp.lang.ada
Subject: Re: Librairies d'IA
Date: Tue, 09 Oct 2007 15:22:37 −0000

There is also FannAda (https://sourceforge.net/projects/lfa/), Ada interface to the Fann (Fast Artificial Neural Network) library.

Fuzzy sets for Ada

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 9 Sep 2007 21:22:47 +0200
Subject: ANN: Fuzzy sets for Ada v5.0
Newsgroups: comp.lang.ada

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

The current version includes distributions of string edit, interval arithmetic and simple components packages. It provides implementations of:

- Confidence factors with the operations not, and, or, xor, +, *;
- Classical fuzzy sets with the set-theoretic operations and the operations of the possibility theory;
- Intuitionistic fuzzy sets with the operations on them;
- Fuzzy logic based on the intuitionistic fuzzy sets and the possibility theory;
- Fuzzy numbers both integer and floating-point ones with conventional arithmetical operations;
- Dimensioned fuzzy numbers;
- Fuzzy linguistic variables and sets of linguistic variables with operations on them;
- Dimensioned fuzzy linguistic variables and sets;
- String-oriented I/O is supported;

The software is distributed under GM GPL.

New in this release:

A rich GUI interface based on GTK+ (The GIMP Toolkit portable across many operating systems including Windows and Linux). The provided set of widgets and tree view cell renders covers:

- Indication and editing of truth values;
- Editing and viewing of fuzzy sets, intuitionistic fuzzy sets and classification in textual and as a list;
- Editing and viewing sets of linguistic variables.

The linguistic variables sets editor features:

- Domain set view representing individual membership functions of the variables;
- Annotated axes of the domain view;
- Dimensioned domains support;
- Scroll bars of the domain view axes;
- Multiple selection of the variables and individual points of membership functions;
- Visual selection of the variables and points of their membership functions in the domain;
- Indication of the selected variables;
- Indication of the selected points of the membership functions of;
- Searching for the points of the membership functions;
- Indication of an accumulated set of the linguistic variables;
- Editing of the accumulated sets;
- Zooming the widget along its axis;
- Zooming in and out per selection of a rectangular area;
- Scrolling the widget;
- Undo/redo buffer for editing;
- A separate undo/redo buffer for all actions changing the visual appearance of the widget;
- Tracking the mouse cursor in the widget;
- Indication and editing the names of the variables in the set in a tree view;
- Indication and editing the points of the membership functions of individual variables in the tree view;
- Checking names for legality and duplication, indication of illegal names;
- Moving groups of selected points of the membership functions along the axis per mouse;
- Adding, removing, moving variables in the set;
- Adding, removing points of the variables;
- Applying operations, such as not, and, or, xor to the selected variables and inserting the result of.

[See also “Updates for Fuzzy sets for Ada, and Simple components” in AUJ 27-2 (Jun 2006), p.72. —su]

Units of measurement for Ada

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 9 Sep 2007 19:51:11 +0200
Subject: ANN: Units of measurement for Ada v2.5
Newsgroups: comp.lang.ada
Organization: cbb software GmbH

The library provides tools for handling dimensioned values in Ada. Checks are run-time when not removed by the compiler. String I/O and GTK+ widgets based on GtkAda included.

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

Changes to the version 2.4:

The procedure Get was added for parsing measures in the form of a numeral multiplied by dimensioned scale. The syntax of input is same as in other procedures. When possible this procedure tries to extract the numeral part and the scale. It also determines whether the input has any dimension in it.

[See also same topic in AUJ 28-2 (Jun 2007), pp.73–74. —su]

Finite element analysis in Ada

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Tue, 06 Nov 2007 21:25:01 +0100
Subject: Re: An open source system for finite element analysis in Ada
Newsgroups: comp.lang.ada

> I am looking for an open source system for finite element analysis written in Ada but it seems that no body has written it yet. Any hint?

> However, if it really doesn't exist then I will write one.

There is a partial translation of a Finite Element Kernel there http://homepage.sunrise.ch/mysunrise/gdm/gsoft.htm#mathpaqs
The "classic" elements are implemented: in 1D, 2D, 3D for linear, square and cubic basis functions.

[See also "Number crunching in Ada" in AUJ 28-3 (Sep 2007), pp.149–150. —su]

AdaOpenGL and AdaMultimedia

From: Alexis Muller <xaelis@free.fr>
Date: Wed, 19 Sep 2007 21:11:27 +0200
Subject: AdaOpenGL and AdaMultimedia
Newsgroups: comp.lang.ada

[...] I have worked on the AdaOpenGL binding. The idea is to use Ada typing instead of C style (naming convention and const int). I have also translated in Ada lot of samples for the OpenGL book. You can get this work from the project's CVS (http://sourceforge.net/projects/adaopengl/), but I need to finish it before doing a new release.

I have also startd a new project : AdaMultimedia (http://sourceforge.net/projects/adamultimedia/). The goal is to provide a library to handle multimedia files. Like ffmpeg but in Ada ;) For now it can decode BMP pictures, AVI containers and the Cinepak codec. I am working on jpeg. [...]

From: Alexis Muller <xaelis@free.fr>
Date: Wed, 19 Sep 2007 21:33:03 +0200
Subject: Re: adaopengl et adamultimedia
Newsgroups: fr.comp.lang.ada

[Translated from French. —su]

> What is the difference to the GtkAda OpenGL part?

The first difference is that this binding is independent. Then, if I am not mistaken, the GtkAda OpenGL part is a direct translation of C to Ada.

The purpose of my changes is to replace the C style constructions by Ada constructions. Substituting for example, the enumerated types by "real" Ada types, such as:

```
type PrimitivesType is
      (Points, Lines, Line_Loop,
       Line_Strip, Triangles,
       Triangle_Strip, Triangle_Fan,
       Quads, Quad_Strip, Polygon);
```

Or to use array types instead of pointers…

[See also "Ada and OpenGL" in AUJ 25-4 (Dec 2004), p.186. —su]

GTKAda contributions

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 9 Sep 2007 19:28:05 +0200
Organization: cbb software GmbH
Subject: ANN: GtkAda contributions v1.8
Newsgroups: comp.lang.ada

New in this version:

- GLib.Object.Ref_Count function was added to obtain GTK object's reference count;
- Get_Tooltips was added to Gtk_Style_Button;
- The package Gtk.Main.Router.GNAT_Stack provides tracing with symbolic traceback of the call stack and exception propagation based on GNAT functionality.

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

From: Maxim Reznik <reznikmm@gmail.com>
Date: Mon, 08 Oct 2007 03:15:55 −0700
Subject: Announce: Binary package for GtkAda-2.10.0 for Windows
Newsgroups: comp.lang.ada

After unsuccessful looking for binary distribution of last version of GtkAda I built it by myself.

The result is on

http://www.ada-ru.org/win_bin_en

named GtkAda-GPL-2.10.0-r2.exe

There are some other binary packages for MS Windows such as ASIS, Glade (DSA), PolyORB (and may be more in the future) there.

The package contains GtkAda, Gtk+ DLL-s built from sources, locale files, examples, docs and patches used in compilation.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Organization: cbb software GmbH
Date: Sat, 13 Oct 2007 15:19:45 +0200
Subject: ANN: GtkAda 2.10 support
Newsgroups: comp.lang.ada

GtkAda contributions

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

units of measurement for Ada

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

fuzzy sets for Ada

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

are now using GtkAda version 2.10.

An unofficial binary release for Windows can be found at http://www.ada-ru.org/win_bin_en. For Linux it is officially available in sources at https://libre2.adacore.com, and can be routinely compiled from.

[See also same topic in AUJ 28-3 (Sep 2007), pp.137–138. —su]

QtAda binding

From: Vadim Godunko <vgodunko@gmail.com>
Newsgroups: comp.lang.ada
Subject: Announce: QtAda 0.2.0
Date: Fri, 28 Sep 2007 16:58:41 −0000

We are pleased to announce a new release of QtAda 0.2.0. This is a major feature release. It includes:

- support for MS Windows/MinGW (binary packages will be available soon)
- bindings to QAbstractPrintDialog, QAbstractProxyModel, QColor, QDockWidget, QKeySequence, QListWidget, QMessageBox, QPrintDialog, QSyntaxHighlighter, QTabBar, QTabWidget, QTextEdit and QTextFrame classes
- extended support for QPolygonF, QPainterPath, QPainter, QGraphicsScene classes
- main window subclassing and dock widgets use example (examples/main_windows/dock_widgets)
- many memory leaks fixed
- new implementation of constructors for "primitive" classes avoids unnecessary memory allocation/deallocation cycles (around 27%) and copy operations

QtAda can be downloaded from SourceForge site:

http://sourceforge.net/projects/qtada/

QtAda is an Ada 2005 language bindings to Qt 4.2 and Qt 4.3. It allows you to easily create powerfull graphical user interface in Ada. QtAda uses a native

thread safe signal/slot mechanism, provides access to more than 120 Qt classes, provides an Ada-aware meta object compiler, supports the development of custom widgets and Qt Designer's custom widget plugins, supports loading at runtime GUI forms from Qt Designer's UI files and so on.

From: Vadim Godunko <vgodunko@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: Announce: QtAda 0.2.0
Date: Fri, 05 Oct 2007 18:14:44 −0000

Andry Ogorodnik has contributed MS Windows binary packages for QtAda!

See QtAda download page on SourceForge site:

http://sourceforge.net/projects/qtada/

From: Vadim Godunko <vgodunko@gmail.com>
Newsgroups: comp.lang.ada
Subject: Annonce: QtAda 1.0.0
Date: Tue, 06 Nov 2007 06:24:33 −0000

We are pleased to announce a new version of QtAda 1.0.0. QtAda is an Ada 2005 language bindings to the Qt framework. It supports most Qt classes from QtCore, QtGui, QtSql, QtDesigner and QtUiTools modules.

QtAda allows you to easily develop cross-platform powerful graphical user interfaces completely in Ada 2005. QtAda programs will have native look and feel on most popular platforms — Microsoft Windows, Mac OS X and Linux/Unix — without any platform specific code. It also allows to use Qt Designer for rapid visual GUI development.

For additional information see QtAda official site: http://www.qtada.com/

[See also same topic in AUJ 28-3 (Sep 2007), p.137. —su]

Hibachi official Eclipse Open Source Project

From: Tom's Hibachi musings
Date: November 5, 2007
Subject: On our way
RSS: http://hibachitom.blogspot.com/2007/11/on-our-way

Hibachi is officially an Eclipse Open Source Project. Since it was approved by the Eclipse Management Organization, we've gotten our newsgroup, mailing list and website set up. We've also gone over the code and adapted it so that it will pass Eclipse IP inspection, including swapping our parser generator for another one.

Since June, I've been trying to build a diverse community (vendors, open source projects, universities, third parties, individuals), and while it took a bit longer than anticipated, I think the effort will be worth it in the long run.

I've presented Hibachi at a couple of conferences (Ada Europe, Ada UK), and this week I'm giving a Hibachi workshop at SIGAda with a couple of the committers helping out.

I also attended the Eclipse Summit in Ludwigsberg, Germany. It was smaller in size than EclipseCon, which definitely has it's advantages in terms of interactions and access.

So now we're just waiting for the Eclipse legal department to give us the ok to put the sources on our server during the parallel IP process. Then we're on our way.

From: Adam Haselhuhn <adam.haselhuhn@aonix.com>
Date: Wed, 24 Oct 2007 15:52:00 −0500
Subject: Re: Update on schedule?
Newsgroups: eclipse.tools.hibachi

> Any further update on the schedule for providing the initial source code for the Hibachi project?

We were using ANTLR version 2 to generate our Ada language parser, but were required to update to v3. We are now working on starting the parallel IP process, which will allow us to commit the initial code the the Eclipse servers. I'll send another update when that process is complete.

Adam Haselhuhn

Hibachi Committer

[See also “Aonix — Eclipse Hibachi Project Unites Ada Suppliers in Common Environment” in this issue, “Hibachi status” in AUJ 28.3 (Sep 2007) p.138 and “Hibachi — Eclipse Ada Development Tools” in AUJ 28-2 (Jun 2007) pp.81–84. —su]

MKUtils — 4NT and Take Command plugin

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: Write 4NT / Take Command plugins with Ada
Newsgroups: comp.lang.ada
Date: Tue, 30 Oct 2007 20:47:54 +0100

I just wanted to let you know that I successfully ported the demo plug-in to Ada. It worked quite smoothly and so I even thrown in a multi threading demo for good measure.

All on all it was quite easy — and certainly easier the the Delphi version as Ada features full fledged wide character support so all that “WideCharToString” and “StringToWideChar” wasn't needed.

If you want to know more visit the JPSoft Wiki:http://www.jpsoftwiki.com/wiki/index.php?title=Ada_Demo_%28plugin%29

Or download the demo from the Project homepage:

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

From: Martin Krischik <krischik@users.sourceforge.net>
Newsgroups: comp.lang.ada
Subject: Re: Write 4NT / Take Command plugins with Ada
Date: Wed, 31 Oct 2007 13:06:12 +0100

> Could you please provide some context for your post?

From the Web-Site:

MKUtils is currently under development. Once it is finished it will be a Plug-in for 4NT and Take Command.

Currently a port of the demo plug-in to Ada is available. It is perfect if you want to learn how to develop plug-ins for 4NT or Take Command.

Under development is a ACL (access control list) support and a trace feature.

Unlike other 4NT and Take Command this Plug-in is open source so you can have a look how it is done. You can also ask to join the project if you have ideas of your own.

OK, unless you use 4NT or Take Command [1] the plug-ins are of little use. Apart perhaps from educational purpose. One could learn the following Ada skills: Stand-alone-library, interfacing with C and usage of Win32Ada.

[1] http://www.jpsoft.com

An operating system design

From: Archeia
Date: 25 Nov 2007
Subject: An operating system design
RSS: http://www.archeia.com/article-1196001779.html

Conversations in #Ada on IRC yesterday turned back to developing an OS in Ada, this has actually been a goal of mine for some time. I had written a demo hello world style kernel in Ada a few years ago, so I thought I'd try to recompile it using the newest version of FSF GNAT; this failed to build.

After looking through the manuals and finding pragma Restrictions, I stripped out all the other pragmas I had in there to reduce the runtime and tried it out using these pragma restrictions; all compiled ok and it even booted on QEMU!

So, I decided to try and put together a high level design document based on thoughts that I've had over the last 10 years or so, which outlines what I want in an OS. I may even get started on this thing!

[http://www.archeia.com/assets/files/os/highlevel.pdf —su]

Ada-related Products

AdaCore — Support for .NET

From: AdaCore Press Center
Date: Monday September 10, 2007
Subject: AdaCore First to Bring True .NET Integration to Ada
RSS: http://www.adacore.com/2007/09/10/adacore-first-to-bring-true-net-integration-to-ada/

PARIS and NEW YORK — September 10, 2007 — AdaCore, provider of the highest quality Ada tools and support services, today announced that its flagship GNAT Pro development environment is now available for Microsoft's® .NET Framework. GNAT Pro's launch on .NET broadens AdaCore's expanding portfolio of Microsoft platforms, which already includes releases for Windows 2000®, Windows 2003®, Windows XP®, and Windows Vista®. Now users of all major Microsoft platforms can also reap the productivity and reliability gains enabled by the Ada language from within the .NET Framework. This especially benefits those creating mission-critical applications, across a broad range of domains, including communications, financial software, and other enterprise systems.

AdaCore's GNAT Pro launch on .NET was specifically designed to meet the demands of the growing .NET user base around the globe. It is the first commercial Ada tool to support the .NET Framework and API — not simply through "unmanaged" (Windows) code, but also through managed .NET code. The product includes an Ada compiler (which supports the new Ada 2005 standard as well as previous versions of the language), a comprehensive toolset, and supplemental libraries and bindings. Through GNAT Pro, developers can build pure Ada applications as well as Ada components in multi-language systems.

GNAT Pro includes specific features that bring together the strengths of the Ada language and .NET. All .NET APIs can be used directly from Ada through an automated binding tool that saves time and enables the re-use of .NET components. Additionally, since .NET integrates smoothly with Microsoft Visual Studio®, developers can now use this familiar IDE to directly edit Ada code.

"The growth of .NET is making it a major platform for developers across a whole range of environments from desktop to embedded devices," commented Arnaud Charlet, .NET Project Manager, AdaCore. "The launch of GNAT Pro for Microsoft .NET is a critical part of our commitment to make Ada a development language of choice on Microsoft platforms, allowing users to benefit from the strengths of both working together."

With GNAT Pro, developers will now be able to take advantage of the complementary features of Microsoft .NET and Ada. GNAT Pro's implementation of Ada 2005 is especially helpful, since the language's new object-oriented features are ideal for interfacing with the .NET API. The security constraints within .NET also closely align with the checks within Ada.

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 new 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 (Ada, C, C++) 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.

AdaCore — GNAT Pro High-Integrity Edition for Servers

From: AdaCore Press Center
Date: Tuesday September 18, 2007
Subject: AdaCore Announces GNAT Pro High-Integrity Edition for Servers
RSS: http://www.adacore.com/2007/09/18/adacore-announces-gnat-pro-high-integrity-edition-for-servers/

BOSTON, Mass., September 18, 2007 — Embedded Systems Conference — AdaCore, provider of the highest-quality Ada tools and support services, today announced its second High-Integrity Edition product GNAT Pro High-Integrity Edition for Servers. The High-Integrity Family was first introduced in June of 2007 with the release of High-Integrity Edition for DO-178B, a product that supports safety-critical avionics and similar standards used in embedded software development. The new GNAT Pro High-Integrity Edition for Servers is fashioned to directly support DO-278, ESARR 4 and 6 and CAP670/SW01, the US, European and UK ground Air Traffic Management (ATM) safety-critical standards. Its capabilities can also be used to support any safety-critical, mission-critical or high reliability system requirements to run on a native platform.

"AdaCore recognizes that our customers require off-the-shelf solutions to meet high-reliability and safety-critical software development standards," said Robert Dewar, President of AdaCore. "They are relying on us to supply support for embedded safety-critical software development. We are now offering this same support for developers needing higher reliability for native platform development."

About GNAT Pro High-Integrity Edition for Servers

GNAT Pro High-Integrity Edition for Servers is an enhanced version of the GNAT Pro technology, designed for building safe and secure software. Its many features help to reduce the cost of developing and certifying systems that have to meet native platform safety standards. The package includes a full multi-language compile system, a configurable Ada run-time library, and integration with best-in-class test capabilities. The run-time library for the GNAT Pro High Integrity Edition for DO-178B has been certified to the highest safety level for DO-178B, Level A, as a part of multiple avionics systems. These life cycle artifacts are available with the Server package as well and directly satisfy the DO-278 standard and are accepted for many other safety standards.

A configurable run-time library accompanies the High-Integrity Edition for Servers product. This allows developers to tailor the run-time based on the language features required by their application. It can be configured from a zero-foot-print (ZFP), Cert library proven previously for embedded development, to full Ada depending on the assurance level requirements for the program. In this way, projects can enforce language subsets and reduce the cost of program certification.

The newly released GNATstack static analysis tool is supplied with this edition. GNATstack computes and outputs data on the absolute maximum memory utilization for programs conforming to the ZFP or Cert subset program libraries. Maximum program memory utilization reports are a common requirement for many safety-critical standards. The GNAT Pro High-Integrity Edition for Servers provides direct support for this requirement.

AdaCore is now leading the industry by offering both native and embedded safety-

critical software development solutions. This product is available for the primary Unix platforms used in Air Traffic Management development such as: Linux, IBM/AIX, HP-UX and Sun/Solaris. It will be made available on other platforms as the market demands. Look for further expansion of the High-Integrity Edition Family with new safety and security solutions in the future.

[See also "AdaCore — GNAT Pro for DO-178B" in AUJ 28-3 (Sep 2007), pp.138–139. —su]

AdaCore — GNATstack 1.1.0

From: AdaCore Developer Center
Date: Thursday September 20, 2007
Subject: GNATstack 1.1.0
RSS: http://www.adacore.com/2007/09/20/gnatstack-110/

AdaCore today announced the availability of GNATstack 1.1.0 to GNAT Pro High-Integrity customers. GNATstack is a static analysis tool that computes the maximum stack usage for a program.

From: AdaCore Press Center
Date: Tuesday September 18, 2007
Subject: AdaCore Announces Innovative Stack Analysis Tool
RSS: http://www.adacore.com/2007/09/18/adacore-announces-innovative-stack-analysis-tool/

BOSTON, Mass. — September 18, 2007 — Embedded Systems Conference — AdaCore, provider of the highest quality Ada tools and support services, today announced the immediate availability of GNATstack, a static analysis tool that helps developers predict the maximum stack usage requirements for their applications. GNATstack is available separately or as part of AdaCore's GNAT Pro High-Integrity Edition products, supporting development for DO-178B, DO-278 and other related safety-critical standards.

The GNATstack tool statically calculates the maximum stack space required by each task in an application. The computed bounds can be used to ensure that sufficient space is reserved, thus guaranteeing safe, predictable execution with respect to stack usage. GNATstack uses conservative analysis to deal with complexities such as subprogram recursion, while avoiding unnecessarily pessimistic estimates. The tool's output data can be used directly to satisfy DO-178B requirements (Table A-5, Objective 6, which relates to the Accuracy and consistency issues itemized in Section 6.3.4f) and the associated sections from DO-278 for native safety systems.

"AdaCore always strives to reduce the cost of software development for our customers," said Robert Dewar, President of AdaCore. "GNATstack can save a large amount of time and effort by proving what an application's maximum memory requirements will be. This information can be used to select the appropriate hardware platform. It can also be used to prove that the selected platform will meet the application's memory requirements, which is required by DO-178B and similar safety standards where exhausting available memory is not an option."

GNATstack exploits data generated by the compiler to compute worst-case stack requirements. It performs per-subprogram stack usage computation combined with control flow analysis. GNATstack is a static analysis tool in that its computation is based on information known at compile time. Thus when the tool indicates that the result is accurate, the computed bound can never be exceeded.

On the other hand, there may be situations in which the results will not be accurate (the tool will indicate such situations) because of some missing information (due to subprogram recursion, indirect calls, etc.). AdaCore provides the infrastructure to allow users to specify this missing call graph and stack usage information.

GNATstack's main output is the worst-case stack usage for every entry point, together with the paths that lead to these stack needs. The list of entry points can be automatically computed (all the tasks, including the environment task) or can be specified by the user (a list of entry points or all the subprograms matching a certain regular expression).

Price and Availability

GNATstack is available with all GNAT Pro High-Integrity Edition Family products. Please contact AdaCore (sales@adacore.com) for the latest information on pricing and supported configurations.

[See also "AdaCore — GNATstack" in AUJ 27-4 (Dec 2006), p.205. —su]

AdaCore — GPRbuild

From: AdaCore Press Center
Date: Tuesday September 25, 2007
Subject: AdaCore Launches New GPRbuild Tool To Speed Multi-Language Development
URL: http://www.adacore.com/2007/09/25/gprbuild/

Easy to use tool aims to solve increasing project complexity

MANCHESTER, UK — Ada Conference UK — September 25, 2007 — AdaCore, provider of the highest quality Ada tools and support services, today announced GPRbuild, an advanced new software tool designed to help automate the construction of multi-language systems. The first tool of its kind, it removes complexity from multi-language development by allowing developers to quickly and easily compile and link software written in a combination of languages, including Ada, assembler, C, C++ and Fortran. Easily extendable by users to cover new toolchains and languages, it is primarily aimed at projects of all sizes organized into subsystems and libraries, and is particularly well-suited for compiled languages.

Designed to work with any version of AdaCore's GNAT Pro development environment, GPRbuild features a built-in knowledge base that understands the characteristics of compilers across a wide variety of languages. Rather than having to maintain complex rules for building each component of a system, developers simply specify the sources' location and compiler options. GPRbuild automatically manages their integration into a complete system.

"With more and more projects becoming multi-language there is a growing need for a generic build utility that can make integration a less complex and more automatic process," commented Cyrille Comar, managing director, AdaCore. "Developed following customer feedback, GPRbuild is simple and straightforward to operate, enabling users to concentrate on development rather than needing to worry about bringing multi-language applications together into a single system."

Previously developers needed to manually link components of multi-language applications, which relied on in-depth knowledge of each compiler involved as well as how they interact with each other. This process had to be repeated if new versions of compilers were used as there was no simple way of collecting and sharing integration data. GPRbuild's inherent knowledge base covers most commonly used toolchains and languages, but also allows developers to add their own information to standardize native and cross-platform configuration.

GPRbuild manages a three-step build process — compilation, post-compilation (binding) and linking.

Compilation:

Each compilation unit of every subsystem is examined in turn, checked for consistency, and compiled or recompiled when necessary by the appropriate compiler. The recompilation decision is based on dependency information usually automatically captured by a former compilation.

Post-compilation (binding):

Compiled units of a given language are passed to a language-specific post-compilation tool where it exists. It is also during this phase that objects are grouped into static or dynamic libraries as specified.

Linking:

All units or libraries from all subsystems are passed to a linker tool specific to the set of toolchains used together.

GPRbuild takes as its main input a project file defining the build characteristics of the system under construction, such as: which sources to use, where to find them, where to store the objects produced by the compiler, and which options the various tools should be invoked with.

While GPRbuild is generic in the sense that it provides equivalent build features for all supported languages, it also allows the addition of new languages and new toolchains and provides a means of configuring aspects, including:

- language characteristics (such as source naming conventions)
- toolchain characteristics (such as compiler invocation)
- subsystem characteristics (such as compiler default options)
- source file characteristics (such as file specific compilation options)

Availability

GPRbuild is immediately available from October 2007 in beta, as part of the GNAT Pro subscription. It runs on all GNAT Pro supported configurations — please contact AdaCore (sales@adacore.com) for the latest information on pricing.

AdaCore — GPS 4.1.3

From: AdaCore Developer Center
Date: Thursday September 20, 2007
Subject: GPS 4.1.3
RSS: http://www.adacore.com/2007/09/20/gps-413/

AdaCore recently announced the availability of GPS 4.1.3 for the x86-windows host. The 4.1.3 version is a follow up release providing mainly fixes to the 4.1 technology. GPS 4.1.3 is compatible with GNAT Pro versions 3.16a1 up to 6.1.0w, and is now fully compatible with Windows Vista. For a full list of new features in GPS 4.1, please [go to http://www.adacore.com/2007/03/28/gps-410/ —su].

[See also "AdaCore — GPS 4.1.0" in AUJ 28-2 (Jun 2007), p.77. —su]

AdaCore — GNAT Programming Studio IDE 4.2

From: AdaCore Press Center
Date: Tuesday November 6, 2007
Subject: Introducing New Version of GNAT Programming Studio IDE
RSS: http://www.adacore.com/2007/11/06/introducing-new-version-of-gnat-programming-studio-ide/

FAIRFAX, Va., November 6, 2007 — SIGAda 2007 — AdaCore, provider of the highest quality Ada tools and support services, today announced the upcoming release of GNAT Programming Studio (GPS) 4.2. This Ada-oriented IDE (Integrated Development Environment) accompanies AdaCore's GNAT Pro toolset on most platforms.

With an emphasis on helping developers improve the quality and maintainability of their software, this GPS release includes a number of new features, enhancements, and plug-ins, including support for code coverage and improved generation of documentation. The advanced code coverage feature enables developers to test code and ensure that the software is properly covered, with the GPS user interface allowing them to visualize the coverage information at different levels of detail (from project down to source lines of code). The improved and faster documentation generator uses Web 2.0 technologies to generate HTML pages, thus providing a web-based view of projects.

"As software complexity increases, developers are looking for an environment to help them create high quality code that is straightforward to produce and understand," commented Arnaud Charlet, GPS Project Manager at AdaCore. "GPS 4.2 delivers these capabilities. With its many enhancements based on customer feedback, it continues to be the environment of choice for professional Ada development."

While GPS already supports the Ada 2005 standard, the GPS 4.2 enhancements will help developers take full advantage of both the new features as well as the existing Ada 95 object-oriented programming facilities. Furthermore, GPS includes multi-language support (including Ada, C, and C++) and is available on a wide range of host environments. Through its intuitive, unified visual interface, developers can access tools from both AdaCore and third parties, easing both development and maintenance. 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, Java, and C++
- 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 and 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
- Copying of text with line numbers prepended
- The ability to close all editors

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

About GPS

GPS is a powerful Integrated Development Environment (IDE) written in Ada, based on the GtkAda toolkit. GPS' extensive source-code navigation and analysis tools can generate a broad range of useful information, including call graphs, source dependencies, project organization, and complexity metrics. It also provides support for configuration management through an interface to third-party Version Control Systems, and supports a variety of platforms, including Altix Linux, IA64 HP Linux, Solaris (sparc and x86), GNU/Linux (x86 and x86-64), and x86 Windows (2000, 2003, XP and Vista). GPS is highly extensible; a simple scripting approach enables additional tool integration. It is also customizable, allowing programmers to specialize various aspects of the program's appearance in the editor for a user-specified look and feel.

Pricing and Availability

GPS 4.2 is scheduled for release at the beginning of December 2007, when it will be available to GNAT Pro customers on selected platforms. GPS is included with the GNAT Pro Ada Development Environment. Please contact AdaCore (sales@adacore.com) for the latest information on pricing and supported configurations.

Aonix — ObjectAda RAVEN for VxWorks 653

From: Aonix Press Releases
Date: September 18, 2007
Subject: Aonix Releases Next-Generation Safety-Critical Platform
URL: http://www.aonix.com/pr_09.18.07b.html

ObjectAda® RAVEN™ supports Wind River VxWorks® 653

Embedded Systems Conference, Boston — September 18, 2007 — Aonix®, a provider of solutions for safety- and mission-critical applications, announced the release of ObjectAda RAVEN for

Windows targeting the Wind River VxWorks 653 multi-partition RTOS for PowerPC. ObjectAda RAVEN features an enhanced safety-critical Ada runtime that communicates to the underlying RTOS through the ARINC-653 APEX interface available in the Wind River VxWorks 653 environment — enabling high execution and safety certification efficiency.

ObjectAda RAVEN is an embedded Ada development system that enables engineers to isolate and build applications for deployment in one or more VxWorks 653 execution partitions. This ability to segment code into separate execution partitions has significant impact to the cost of safety-critical certification since developers can separate the code subject to the highest levels of criticality from less-critical code and thereby separately test and scrutinize it for the appropriate level of certification. In addition, Wind River VxWorks 653 includes DO-178B Level A qualified development tools that further reduce the testing burden of updated code and modified configurations. With standard industry costs to create certification evidence starting at $45 and often creeping into the hundreds of dollars per application source line, any reduction or separation of lower-criticality code yields dramatic savings.

ObjectAda RAVEN for Windows targeting VxWorks 653/PowerPC consists of a fully compliant ACATS 2.5 Ada 95 compiler with supporting tools including a build/bind tool, library tool and debugger, and delivered with a predefined program library which is based on the Ravenscar profile subset of the full predefined language. The Ravenscar profile, adopted at the Eight International Real-Time Ada Workshop (IRTAW-8), Ravenscar UK, and subsequently made part of the Ada 2005 specification, accommodates certification requirements for high-integrity (safety-critical) real-time systems. The ObjectAda RAVEN runtime environment is certifiable to DO-178B Level-A while supporting task execution through the underlying VxWorks 653 RTOS and board support services.

"The Aonix ObjectAda RAVEN product's integration with Wind River VxWorks 653 gives the aerospace and defense industry a very powerful platform," noted Chip Downing, Senior A&D Industry Marketing Manager at Wind River. "The Aonix Ada runtime combined with the Wind River VxWorks 653 platform make the ObjectAda RAVEN solution highly portable and very appealing to safety-critical developers needing to deploy the latest industry standards."

"Support for Wind River's world-class time and space partitioned RTOS is taking Aonix to a new level in the safety-critical space," said Gary Cato, director of strategic alliances at Aonix. "Certifying applications is expensive in both human capital investment and time to market. By providing developers with products that improve code reliability, reduce testing overhead and shorten certification cycles we help customers realize significant savings." Aonix ObjectAda products give developers the choice between the traditional Aonix IDE for development and the new AonixADT™ Eclipse plug-in. Developers also gain full access to the Wind River Workbench environment including Ada debug and the ability to debug across multiple languages.

Shipping and Availability

ObjectAda Real-Time RAVEN for Windows targeting VxWorks 653/PowerPC is immediately available. Prices range from $15,000 to $25,000 for a single seat license depending on bundle options. Certification evidence is available based on demand and will be priced at that time. 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 PikeOS" in AUJ 28-1 (Mar 2007), pp.12–13 and "Aonix — ObjectAda RAVEN for PowerPC" in AUJ 28-1 (Mar 2007) p.13. —su]

Aonix — Eclipse Hibachi Project Unites Ada Suppliers in Common Environment

From: Aonix Press Releases
Date: November 5, 2007
Subject: Eclipse Hibachi Project Unites Ada Suppliers in Common Environment
URL: http://www.aonix.com/pr_11.05.07.html

Aonix, DDC-I, CohesionForce, and other suppliers providing industry support

SIGAda Conference, Fairfax, VA, November 5, 2007

The Eclipse Foundation today announced the creation of a new open-source project called Hibachi. The Hibachi project provides an industrial-strength, vendor-neutral Ada integrated development environment (IDE) that also serves as a platform for other contributors to provide value-added functionality for Ada developers. Hibachi is a sub-project of the Eclipse Tools Project, and it parallels and complements CDT, the C/C++ Development Tooling project, providing a multi-language native embedded software development environment. The name Hibachi is an anagram honoring the late Jean Ichbiah, lead designer of the Ada language.

To initiate the project, Aonix has contributed the source code of AonixADT, an existing commercial Eclipse plug-in technology that supports Aonix ObjectAda as well as GNAT tool chains on a variety of host and target platforms, as the initial code for the project. AonixADT is based on JDT and CDT, the Java and C Eclipse development toolkits. Additional contributions to Hibachi are being actively solicited by the project team.

Tom Grosman of Aonix was selected as project lead, supported by Adam Haselhuhn of Aonix, Lisa Jett of DDC-I, Mandy McMillion and David Philips of CohesionForce, and other industry participants. Other organizations planning to contribute to Hibachi include OC Systems, Praxis High Integrity Systems, existing opensource Ada projects, as well as universities and interested individuals. The Hibachi Project is mentored by CDT Project Lead Doug Schaefer of QNX and DSDP Lead Doug Gaff of Wind River.

"The Eclipse Hibachi project will promote wider adoption of Eclipse-based development by the Ada community, which includes many major high-integrity projects worldwide," said Mike Milinkovich, executive director of the Eclipse Foundation. "Formally adopting Ada functionality into Eclipse will encourage easier integration of Ada development alongside other development tools and language platforms supported by Eclipse. Eclipse provides an ideal solution, giving Ada developers a universal open-source platform with a broad ecosystem of plug-ins."

"Aonix is excited to play a central role in Hibachi and to extend our involvement in the Eclipse community for the benefit of our customers and Ada users in general," said Dave Wood, Aonix VP marketing. "For years, we have been committed to Eclipse solutions for the benefit of our Java and Ada customers, and our ability to provide proven sources and project leadership to help launch the Hibachi project represents the next stage of our commitment." Major Hibachi functionality includes:

- Ada editor with semantic navigation, code assist, structural representations, and formatting
- Build configurations
- Debugging support
- Refactoring
- Support for multiple tool chains

- Native or embedded launch capability
- Wizards and templates

The Hibachi project aims to become the benchmark Ada IDE, by which all other Ada environments are measured, and the first choice for Ada developers. Functionally, Hibachi will shadow the ongoing development evolution of CDT. The first year development will focus on supporting multiple Ada compiler technologies, offering closer evolution with the CDT architecture, providing useful and stable APIs, and integrating with the Eclipse DSDP/TM and DSDP/DD projects. Subsequent phases will emphasize implementation of new and improved functionality, such as refactoring and analysis tools, and ever-increasing integration with more varied tools.

Support for toolchain extension points with integrations available from multiple Ada vendors is anticipated early in 2008. Re-architecture work to take advantage of the latest CDT developments and create robust and stable APIs will result in incremental releases in mid-2008, and the first major version (v1.0) is scheduled to take advantage of improvements of DSDP in the Ganymede update later in the year.

In addition, Hibachi will provide an open framework for the integration and use of other tools used during the lifecycle of large-scale Ada application development. These tools include but are not limited to analysis, modeling, testing, verification, documentation, refactoring, and configuration management.

About the Eclipse Foundation

Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software. Eclipse provides extensible tools and frameworks that span the software development lifecycle, including support for modeling, language development environments for Java, C/C++ and others, testing and performance, business intelligence, rich client applications and embedded development. A large, vibrant ecosystem of major technology vendors, innovative start-ups, universities and research institutions and individuals extend, complement and support the Eclipse Platform.

The Eclipse Foundation is a not-for-profit, member supported corporation that hosts the Eclipse projects. Full details of Eclipse and the Eclipse Foundation are available at www.eclipse.org.

[See also "Hibachi official Eclipse Open Source Project" and "DDC-I Joins Eclipse Hibachi Project" in this issue. —su]

DDC-I Joins Eclipse Hibachi Project

From: DDC-I Press Releases
Subject: DDC-I Joins Eclipse Hibachi Project
Date: November 5, 2007
URL: http://www.ddci.com/display_news_item-filename-news_EclipseHibachiProject_release.htm

New Initiative Will Unite Ada Suppliers in Common Environment and Extend Ada Ecosystem

SIGAda Conference, Fairfax, VA, November 5, 2007 — DDC-I, a leading supplier of development tools for safety-critical applications, today announced that it has joined the Eclipse Foundation's new open-source Hibachi project.

DDC-I, a founding member of the Hibachi Workgroup, will work closely with other project members to develop a common Eclipse-based Ada environment that can accommodate Eclipse-based Ada tools, application software, and other plug-ins from multiple suppliers. DDC-I will also work to provide Hibachi compatibility for its Eclipse-based SCORE®-Ada and OpenArbor mixed language development tools, which are optimized for safety-critical applications.

Hibachi is a sub-project of the Eclipse Tools Project. Its mission is to develop an industrial-strength, vendor-neutral Ada integrated development environment (IDE) that provides a benchmark for measuring all other Ada environments. The open Eclipse-based IDE will also serve as a platform for integrating value-added Ada plug-ins from other vendors, thereby enhancing and extending the Ada ecosystem.

"We are very excited to have DDC-I participate and contribute in the Eclipse Hibachi project", said Mike Milinkovich, executive director of the Eclipse Foundation. "It is also an important statement of support for Hibachi that DDC-I will be supporting their commercial products on this platform."

DDC-I is a long-time Eclipse member and a pioneer in the development of standard C, Ada, and Java tools and run-time platforms for safety-critical applications. DDC-I introduced the first real-time Ada debugger, the first validated 1750A Ada compiler, the first FAA-certified multitasking run-time system, the first ANDF Ada 95 compiler, and the first compiler to pass Ada Conformity Assessment Test Suite. DDC-I is also a member of the Safety-Critical Java Expert Group (JSR 302), whose mission is to create a subset of real-time Java suitable for safety-critical applications requiring FAA certification. "DDC-I looks forward to working with the Eclipse and Hibachi community to develop a common Eclipse-based Ada environment that fosters multi-vendor interoperability, simplifies mixed-language development, and makes it easy for safety-critical developers to combine our best-of-breed development tools with top-notch tools from other vendors," said Bob Morris, president of DDC-I. "We are committed to offering the industry's most advanced Eclipse-based compiler and debug technology for developing mixed-language Ada, C, and Java safety-critical applications."

[See also "Hibachi official Eclipse Open Source Project" and "Aonix — Eclipse Hibachi Project Unites Ada Suppliers in Common Environment" in this issue. —su]

DDC-I — OpenArbor Eclipse Development Suite

From: DDC-I Press Releases
Subject: DDC-I Announces Eclipse-Based Mixed Language Development Suite for Real Time Embedded Development
Date: October 15, 2007
URL: http://www.ddci.com/display_news_item-filename-news_Eclipse-BasedMixedLanguage_release.htm

New IDE simplifies mixed C, Embedded C++, Ada and real-time Java development and software migration

Phoenix, AZ. October 15, 2007. DDC-I, a leading supplier of development tools for safety-critical applications, today announced the first Eclipse-based mixed-language development and run-time environment to integrate C, Embedded C++, Ada, and real-time Java. Known as OpenArbor, the new IDE makes it possible to develop hard real-time applications that combine Java, C, EC++, and Ada.

"OpenArbor is the only Eclipse-based IDE that supports true mixed language C, Embedded C++, Ada, and real-time Java development," said Bob Morris, president and CEO of DDC-I. "OpenArbor addresses all aspects of real-time mixed language application development, debugging, testing, and deployment on the target system."

"Mixed language development is becoming increasingly prevalent, particularly for applications requiring the migration of existing code," said Steve Balacco, Director, Venture Development Corp (VDC). "OpenArbor's Eclipse packaging and unified mixed language capability should make it easier for developers to migrate, maintain, and upgrade existing code while utilizing emerging languages like real-time Java for new development projects."

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 EmbeddedC++ with the SCORE multi-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 and Java development. These plug-ins can also be used with popular IDEs such as Wind River Workbench and LynuxWorks Luminosity.

The Ada plug-in, known as SCORE®-Ada, features an optimizing Ada compiler and run-time environment optimized for safety-critical embedded Ada projects. The SCORE-Ada debugger supports full Ada-level debugging, including constraints, attributes, tasking, exceptions, break-on-exception and break-on-tasking events. The debugger is non intrusive, can debug at the source or machine level, and can be enabled without changing the generated code.

OpenArbor's real-time Java plug-in, known as Scorpion, is the only real-time Java that provides deterministic garbage collection, a prerequisite for executing bounded, hard real-time applications. Scorpion features a Java compiler, a builder for ahead-of-time Java file compilation, and a virtual machine (ScorpionVM) for executing real-time Java applications. Scorpion also features a smart linker that reduces code size (up to 80%) by removing unused objects from closed systems, and a profiler that helps optimize speed/size tradeoffs by determining the best mix of compiled and interpreted code.

Scorpion is also available with an Eclipse plug-in that automatically maps Java native method calls directly to existing Ada/C code. This unique tool enables Java programs to call existing C and Ada programs, thereby simplifying mixed language development and the migration of legacy C/Ada code.

OpenArbor provides versatile run-time target options, including a bare run-time system certifiable to Level A of the FCC DO-178B standard, an enhanced bare run-time system for simulated and emulated environments, and popular RTOSes such as Wind River's VxWorks, LynuxWorks LynxOS-178 and Ardence's RTX real-time extensions for Windows.

OpenArbor is available immediately. 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

Lattix — Lattix 3.5

From: Lattix News
Subject: Lattix Releases Lattix 3.5
Date: September 19, 2007
URL: http://www.lattix.com/news/articles/Lattix35.php

Award-winning software architecture management solution now available for complex C/C++ and Ada embedded systems

BOSTON, MA — September 19, 2007 — Lattix Inc., the leading provider of innovative software architecture management solutions, today at the Embedded Systems Conference announced the release of its newest solution, Lattix 3.5. This solution features new C/C++ and Ada modules which enable architects, developers and managers to visualize, test, and maintain the architecture of their complex embedded systems.

Lattix has pioneered the Dependency Structure Matrix (DSM) approach which uses dependencies to create the most accurate and scaleable blueprint of software applications, databases and systems. Lattix 3.5 enables this approach for embedded systems by introducing new C/C++ and Ada modules which integrate with Understand, the popular reverse engineering tool and IDE from Scientific Toolworks.

"We think the DSM technology from Lattix is already incredibly useful for C/C++ projects, said Ken Nelson, president of Scientific Toolworks. "Combining it with Understand's code analysis technology makes it faster and more accurate for very large C/C++ programs."

"Our integration with Understand addresses the need for a more complete and scalable solution for large C/C++ and Ada systems" explains Neeraj Sangal, president and founder of Lattix. "Our customers can now quickly achieve measurable results at any stage of development by understanding and improving the architecture, eliminating rogue dependencies, expediting refactoring efforts, and reducing defects."

In addition to the new C/C++ and Ada modules, Lattix 3.5 provides these unique capabilities:

- Compact DSM visualization of the architecture and dependencies of systems comprised of applications, databases, frameworks, and services
- Powerful DSM algorithms to analyze and specify structure, identify opportunities for refactoring, and perform impact analysis before making changes to the code
- Automatic updating of architectural changes to immediately alert for violations of the architecture and dependency rules

Lattix 3.5 is available immediately with modules for Ada, C/C++, Java, .NET, Oracle, Spring, Hibernate, and LDI. Lattix 3.5 also provides support for full web-based reporting of architectural metrics, violations, and incremental changes. For more information, please visit http://www.lattix.com/products/LDM.php. A free evaluation license is also available for download from http://www.lattix.com/dl/gettingstarted.php.

Lattix 3.5 enables companies to improve and maintain quality, enhance testability, lower costs through more effective development, and manage risks by understanding the impact of proposed changes.

About Lattix

Lattix is the leader of software architecture management solutions that deliver higher software quality and lower risk throughout the application lifecycle. Lattix LDM provides a powerful new approach of utilizing dependency models for automated analysis and enforcement of architectures. Lattix is located in Andover, MA. More information about Lattix can be found at http://www.lattix.com.

RTI — DDS-Compliant Real-Time Middleware

From: RTI News Releases
Subject: Working With Saab, RTI Integrates Support for Ada with DDS-Compliant Real-Time Messaging Middleware
Date: November 12, 2007
URL: http://www.rti.com/corporate/news/saab.html

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

SANTA CLARA, CA — 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 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 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 (Ada, C and C++) 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 confidence 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-response, expert support service.

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 reliability, efficiency and safety are critical. AdaCore's flagship product is GNAT Pro, which comes with expert online support and is available on more platforms than any other Ada technology. AdaCore has customers worldwide; please visit http://www.adacore.com/home/company/customers for more information. Use of Ada and GNAT Pro continues to grow in high-integrity and safety-critical applications, including avionics, defense, air traffic control, railroad systems, financial services and medical devices. AdaCore has North American headquarters in New York and European headquarters in Paris. www.adacore.com.

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.

Ada and GNU/Linux

Debian transition to GCC 4.2

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 10 Oct 2007 00:34:03 +0200
Subject: Ada in Debian: transition to GCC 4.2
Newsgroups: comp.lang.ada

The transition of Debian to GCC 4.2 has started. Debian Unstable deserves its name.

Thanks to massive help from Xavier Grave, gnat-4.2 now provides a version of the Ada run-time using the setjump/longjump exception handling mechanism. This new run-time library is only provided in static form as an alternative to the existing zero-cost exception handling mechanism, which is still provided as both static and shared libraries.

The SJLJ version of the run-time is particularly important for Annex E distributed systems. In addition to his work on gnat-4.2, Xavier Grave has also updated the gnat-glade package to version 2007, using the SJLJ library. His initial testing shows quite an improvement over gnat-glade 2006 using the ZCX mechanism. gnat-glade is almost ready for upload but still needs a little polishing. We will upload it in a few days.

Before gnat-glade, I will upload a new and final gnat-4.1 which no longer provides libgnatprj-dev or libgnatvsn-dev. These packages are now provided by gnat-4.2 instead. Once that's done, a new upload of gcc-defaults will make gnat-4.2 the new default compiler.

After that, I will update and re-upload all Ada packages over the course of several months, as my free time permits.

I intend to work in roughly this order:

gnat-glade (upgrade to 2007 with SJLJ exceptions)
gnat-gdb (upgrade to 6.4+2007)
asis (upgrade to 2006 or 2007, whichever works best)
adacontrol (upgrade to 1.7)
libgtkada2 (upgrade to 2.10)
libtemplates-parser (upgrade to a recent CVS snapshot)
libxmlada2
gnat-gps
libaws
gnade
libflorist
libaunit

libopentoken
libtexttools

If you would like to help, please read this introduction:

English: http://www.ada-france.org/article131.html

French: http://www.ada-france.org/article130.html

[See also same topic in AUJ 28-3 (Sep 2007), p.143. —su]

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 10 Oct 2007 12:00:57 +0200
Subject: Re: Ada in Debian: transition to GCC 4.2
Newsgroups: comp.lang.ada

> I'm using GLADE with ZCX without trouble since long time... So, just curious, what is the relation between SJLJ and Annex-E ? What problem did you have ?

Xavier knows more about this. It has to do with propagating exceptions across partitions; the ZCX mechanism is apparently not supported or very buggy. [...]

> [...] note that AWS comes with the templates_parser engine, so above libtemplates-parser is the standalone version, right?

In Etch, this is a snapshot of the standalone version which also works with AWS and GPS; both use slightly different versions of it. I intend to do the same for Lenny.

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Newsgroups: comp.lang.ada
Subject: Re: Ada in Debian: transition to GCC 4.2
Date: Fri, 12 Oct 2007 02:11:31 +0200

> Templates_Parser and AWS have been moved to Subversion on Libre site since some time now. There is no more CVS repositories for those projects.

Thanks for the info; I had noticed that the CVS server was down last week but not that you had moved some projects to Subversion. Now I see that AdaCore have in fact moved most of the projects to Subversion (the only ones still in CVS seem to be GLADE and GDB).

This has allowed me to make a long-standing dream come true: I have tailored the full history into a Monotone database. This db is private for now, but I can merge it into the Ada-France database if anyone is interested.

The down side is that the Subversion interface no longer allows me to see the tags; this is a result of Subversion's broken working model (tags are directories) and of the fact that the tags directory in the AdaCore repo is private.

References to Publications

AdaCore — GNAT Pro Insider

From: AdaCore Press Center
Date: Monday November 19, 2007
Subject: Nov 2007 Contents
RSS: http://www.adacore.com/2007/11/19/nov-2007-contents/

- Gnat Pro High-Integrity Family Expanding to Servers
- US Navy Policy Recognizes Open-Source Software

- Current Releases
- In the Pipeline
- Spotlighting a GAP Member
- Interview with Gregory Gicca
- Webinar Schedule
- AdaCore Partner Praxis High Integrity Systems Makes SPARK/Ada a Language to Depend on
- Conferences/Events

http://www.adacore.com/wp-content/uploads/2007/11/adacore_news_1107_web.pdf

[See also "GNAT Pro Insider newsletter" in AUJ 28-3 (Sep 2007), p.144 —su]

Embedded System Engineering — "Ada for certified safety-critical systems"

From: AdaCore Press Center
Date: Thursday September 27, 2007
Subject: Ada for Safety-Critical Systems
URL: http://www.adacore.com/2007/09/27/ada-for-safety-critical-systems/

Embedded System Engineering

"Ada for certified safety-critical systems" by Dr. Jose F. Ruiz, Senior Software Engineer, AdaCore.

[See http://www.esemagazine.com/... —su]

GNC — "The 10 percent rule of system upgrades"

From: Tom Panfil <tapanfil@ieee.org>
Organization: ACM SIGAda
Date: Mon, 19 Nov 2007 00:37:18 GMT
Subject: ACM SIGAda 2007 Wednesday Keynote Talk Covered by GCN
Newsgroups: comp.lang.ada

Joab Jackson of Government Computer News attended much of ACM SIGAda 2007 and published a little teaser story based upon the presentation of our second Keynote Speaker. This is a prelude to more extensive coverage planned for the 11 DEC issue. At least one of my photos will probably be used. You can read the teaser at:

http://www.gcn.com/blogs/tech/45407.html

Note that the story includes a link to the SIGAda website, which provides links to both the SIGAda 2007 and 2008 conference sites.

Consider writing a paper or developing an experience report and making a presentation at SIGAda 2008. It is planned to be held in Oregon in October.

Tom Panfil — Registration Chair — ACM SIGAda 2007

[See also "Nov 4–9 — SIGAda 2007 Conference" in this issue —su]

Embedded Real-time Software

From: AdaCore Press Center
Date: Wednesday November 21, 2007
Subject: Embedded Real-time Software (ERTS) 2008
RSS: http://www.adacore.com/2007/11/21/embedded-real-time-software-erts-2008/

4th European Congress ERTS

AdaCore is a major sponsor of this event and Franco Gasperoni will be presenting the paper "Free Software and Leveraged Service Organizations".

Ada-France 2007

From: AdaCore Press Center
Date: Monday November 19, 2007
Subject: Ada France Technical seminar
RSS: http://www.adacore.com/2007/11/19/ada-france-technical-seminar/

Ada France will be hosting a one day technical seminar on the topic of methods, processes, models, and tools for the development of hard real-time embedded applications. The event will be hosted by the ENST Bretagne in Brest, France.

AdaCore is sponsor of this event and Dr. Jose F. Ruiz will be presenting a paper on "Ada 2005 for Real-time Embedded Systems".

For more information, please [http://www.ada-france.org/article137.html]

[See also "Dec 6 — Ada-France 2007" in this issue —su]

Avionics '08

From: AdaCore Press Center
Date: Wednesday November 21, 2007
Subject: Avionics 08
RSS: http://www.adacore.com/2007/11/21/avionics-08/

AdaCore is a major sponsor of this event and will be pleased to meet you on our booth F19.

Michael Friess will be presenting a workshop on "Easing the Development of Certified Avionics Software with Ada".

AdaCore will also be participating in the Wind River led master class “Next Generation Design Airflow”.

Wind River EMEA Aerospace and Defence Seminars

From: AdaCore Press Center
Date: Wednesday September 21, 2007
Subject: Wind River EMEA Aerospace and Defence Seminars
RSS: http://www.adacore.com/2007/09/12/wind-river-emea-aerospace-and-defence-seminars-3/

Wind River EMEA Aerospace and Defence Seminars

AdaCore is Gold sponsor of these events. As such we will be exhibiting and presenting a demo of our toolset.

To register for these events, please [go to http://www.windriver.com/... —su]

[See also same topic in AUJ 28-2 (Jun 2007), p.87. —su]

Ada Inside

Ada helps win Cryptography Challenge

From: Ada Information Clearinghouse
Date: November 2007
Subject: Ada helps win WWII Cryptography Challeng
URL: http://www.adaic.org/news/crypto.html

Ada helps win WWII Cryptography Challenge

Joachim Schueth, a German amateur radio enthusiast from Bonn, won a challenge to crack secret messages encoded by a World War Two cipher.

His program, written in Ada especially for the challenge, cracked the supposedly hardest part of the challenge — deciphering the code of a Lorenz SZ42 encryptor, which has approximately 16 million million million permutations — in just 46 seconds. He completed the entire challenge in less than two hours.

Reuters quoted Andrew Clark, director of Britain's National Museum of Computing, as saying “It's a brilliant piece of work, really really impressive”.

We here at the AdaIC would like to think that his choice of programming language had something to do with his success. Schueth's web site describes Ada as a “powerful and beautiful language [which] has become my favourite”, a sentiment shared by many Ada programmers.

For news reports on the challenge, see German amateur cracks WWII mega-code in 46 seconds [Reuters] and German amateur code breaker defeats Colossus [The Register].

Also see Joachim Schueth's web site on the challenge.

[References:

- Source code: http://www.schlaupelz.de/SZ42/SZ42_software.html

- The Register: http://www.theregister.co.uk/2007/11/16/german_code_breaker_defeats_colossus/

- Reuters http://www.reuters.com/article/mapNews/idUSL1665121720071116

—su]

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Fri, 16 Nov 2007 04:51:11 −0800 (PST)
Subject: Ada helps win WWII Crypto challenge!
Newsgroups: comp.lang.ada

[...] That's got to be worth some marketing mileage to the Ada community!

Well done Joachim, if you're reading this!!

From: Ian Clifton <ian.clifton@chemistry.oxford.ac.uk>
Newsgroups: comp.lang.ada
Subject: Re: Ada helps win WWII Crypto challenge!
Date: 16 Nov 2007 16:09:00 +0000

This was mentioned, and that Ada was used, on the BBC's flagship “Today” news programme this morning.

From: Larry Kilgallen <Kilgallen@SpamCop.net>
Subject: Re: Ada helps win WWII Crypto challenge!
Date: 16 Nov 2007 18:05:07 −0600
Newsgroups: comp.lang.ada

> From the article:

> “Schüth wrote specialist software in the tricky language of Ada....”

> Dunno what to say.

You say “writing in Ada qualifies one as a true expert” :-)

From: Manuel Gomez <mgrojo@gmail.com>
Date: Sat, 17 Nov 2007 06:15:33 −0800 (PST)
Subject: Re: Ada helps win WWII Crypto challenge!
Newsgroups: comp.lang.ada

Choosing well the language has not being his only skill. See this quote from [Reuters:]

“It's a brilliant piece of work, really really impressive,” said Andrew Clark, director of Britain's National Museum of Computing, which designed the challenge and is overseeing the running of Colossus, based at Bletchley Park outside London.

“He's used a program that is highly optimized for this task and he's designed it very well.”

U.S. Air Force T25 SECT Electronic Combat Trainer

From: AdaCore Press Center
Date: Tuesday September 18, 2007
Subject: AdaCore Helps AAI Upgrade the T25 SECT Electronic Combat Trainer
RSS: http://www.adacore.com/2007/09/18/adacore-helps-aai-upgrade-the-t25-sect-electronic-combat-trainer/

BOSTON, Mass. — September 18, 2007 — Embedded Systems Conference — AdaCore, provider of the highest quality Ada tools and support services, today announced another successful deployment of a mission-critical system using its GNAT Pro development environment. AAI Services Corporation utilized GNAT Pro as part of an overall upgrade to the U.S. Air Force T25 Simulator for Electronic Combat Training (SECT) system. The T25 SECT system is a software-based training aid that uses interactive combat laboratory exercises and simulated training missions to teach the principles of electronic countermeasures. As part of the upgrade, AAI Services updated one processor on the system’s student station from an SGI VME-based computer to a single board computer running Windows. The original software for the updated processor was ported to a different host and a new development station was added to the existing Training System Support Center.

AAI Services used GNAT Pro for Windows along with a variety of AdaCore partner software to satisfy the T25 SECT program upgrade demands. AAI Services ported and developed new software using a powerful collection of software libraries that is uniquely available with the GNAT Pro development environment, including:

- GNAT Pro
- OpenGL (Open Graphics Library)
- GLUT (OpenGL Utility Toolkit)
- FreeGLUT
- Win32 Bindings
- Touch screen driver from ELO

“AdaCore is pleased to have provided software solutions to meet the T25 SECT program upgrade requirements,” said Robert Dewar, President of AdaCore. “We strive to offer exceptional software tools, libraries, and services. In addition, we have established relationships with other best-in-class partners that enable us to provide complete solutions for our customers.”

About the T25 SECT System

The T25 SECT provides computer simulations consisting of interactive, electronic combat lab exercises and simulated training missions. It provides a full range of electronic combat (EC)

training from basic threat recognition to complex real-world EC airborne mission scenarios. The T25 SECT trains students in all fundamental aspects of EC, including the operation and utilization of a wide variety of generic, representative EC equipment. SECT provides full-scale mission simulations for typical operations such as strategic/covert penetration, standoff jamming/direct support (SOJ/DS), electronic intelligence (ELINT) collection, and suppression of enemy air defenses (SEAD).

The T25 SECT port required moving one processor in the student station from a VME-based SGI (circa 1993) running IRIX to a VME-based single board computer (SBC) running Windows 2000. The specific platform was a General Micro Systems, Inc. (GMS) V265 Condor SBC that has an Intel Pentium M Processor 1.00 GHz AT/AT Compatible with 515440 KB RAM and an Intel 82852/82855 GM/GME Graphics Controller. The application used SGI's GL calls to accomplish drawing graphics primitives. The GL to OpenGL porting guide was used to translate the GL calls to the Windows implementation of OpenGL drawing primitives.

AAI Corporation also designed and developed the original SECT system for the U.S. Air Force in the 1990s, replacing the AAI-developed Simulator for Electronic Warfare Training (SEWT), which had been in operation since the 1970s.

About AAI Services Corporation

AAI Services Corporation, headquartered in Hunt Valley, MD, is a wholly owned subsidiary of AAI Corporation providing government and commercial customers with responsive, efficient and effective technical services covering a wide variety of technologies and equipment worldwide. AAI Services Corporation's managers, engineers, operators, maintainers, instructors, and logisticians are experts in structuring innovative, performance-based solutions for life cycle support, supply chain management, obsolescence, operational enhancement and skills development for customers' systems, facilities and equipment.

TOPCASED Project

From: AdaCore Press Center
Date: Monday October 29, 2007
Subject: AdaCore Joins TOPCASED Project
RSS: http://www.adacore.com/2007/10/29/adacore-joins-topcased-project/

PARIS and NEW YORK, October 29, 2007 — AdaCore, provider of the highest-quality Ada tools and support services, today announced its participation as a member of the TOPCASED Project (Toolkit in Open Source for Critical Applications & System Development).

Initiated in 2004, the TOPCASED Project brings together a consortium of commercial, scientific, and academic partners, ranging from consulting companies to large industrial groups, including Airbus, Thales, Siemens VDO, CNES, Rockwell Collins, EADS Astrium, CS, Atos, SOPRA, and many more. Its goal is to provide an open-source development environment for producing software that meets the requirements of safety-critical embedded systems. This major cooperative project, which will produce an environment covering the full spectrum of software development, from requirements specification to implementation, is being funded with the help of the French General Business Directorate, or DGE.

The complexity of embedded systems used in the avionics, space and transportation industries places unique demands on development tools, which the traditional off-the-shelf software market alone cannot fulfill. For example, the lifetime of aerospace products can often be as long as 10 to 30 years and, currently, no software products company can commit for such a prolonged period of time at a realistic cost.

To reach the necessary quality levels of safety-critical embedded systems and increase the productivity of developers, system engineering methods and tools need to be improved. By using an open-source approach, the TOPCASED Project hopes to ensure the availability of reliable tools over a long period of time, to share the development costs among all those involved, and to make use of valuable tools already available.

AdaCore is leveraging its experience in safety-critical and embedded software engineering to bring TOPCASED users an advanced Eclipse-based environment for developing Ada applications.

"AdaCore's reputation in the open-source industry meant that we were keen to secure its participation in TOPCASED," said Patrick Farail, Head of Software Development Methods Support, Airbus France. "We felt that AdaCore's approach to embedded development for avionics, its history of working on large, complex and safety-critical projects, and the company's impressive list of commercial and military avionics customers made it the perfect partner for the project," he added.

"AdaCore understands the unique challenges of the embedded aerospace market, and we are excited to be an active member of TOPCASED, working with industrial and academic partners. Our company has a longstanding commitment to open-source, safety-critical, and embedded software development, as evidenced by our support for the Ada community," said Nicolas Setton, AdaCore's TOPCASED Project Manager.

Please see: http://www.topcased.org/ for further details about The TOPCASED Project.

Raytheon's SSDS

From: AdaCore Press Center
Date: Tuesday November 6, 2007
Subject: GNAT Pro Provides Multi-Language Support aboard Raytheon's SSDS
RSS: http://www.adacore.com/2007/11/06/gnat-pro-provides-multi-language-support-aboard-raytheons-ssds/

FAIRFAX, Va., November 6, 2007 — SIGAda 2007 — AdaCore, provider of the highest quality Ada tools and support services, today announced that Raytheon has delivered the Ship Self-Defense System (SSDS) Mk 2 using GNAT Pro for LynxOS within its multi-language software development environment. SSDS Mk 2 is a combat system that integrates and coordinates the sensors and weapons systems aboard a US Naval vessel to provide a coherent tactical picture for situational awareness, command and controls, and quick-reaction self-defense. It is a single-source baseline that supports multiple system configuration modifications (MODs) for large deck ship classes (aircraft carriers and amphibious ships). GNAT Pro was specifically used on the SSDS Mk 2 to support Ada application development on the Intel processors and the LynxOS operating system.

"SSDS Mk 2 is a modular distributed program consisting of C, C++, and Ada software components," said Mark A. Hodge, SSDS Mk 2 Technical Director for Raytheon Integrated Defense Systems. "LynxOS for x86 was selected because of its real-time determinism as we migrated from an older operating system towards a more mainstream OS. The AdaCore GNAT Pro compile system was selected both for its support for LynxOS and its association with GNU, which is being used for the C and C++ application components."

"AdaCore's GNAT Pro tool set provided Raytheon with the seamless interoperability it required to support the inherently mixed-language development of SSDS Mk 2," said Robert Dewar, President of AdaCore. "On large, mission-critical systems, Ada is often used in conjunction with other languages, and the Ada design specifically caters to such usage. AdaCore provides complete support for compilation with multi-language build and debug for all GNAT Pro environments."

SSDS Program Description

On many of today's amphibious ships and aircraft carriers, the radar and anti-air weapons used for self-defense are installed as stand-alone systems. As a result, considerable manual intervention is

required to complete the detect-to-engage sequence against anti-ship cruise missiles (ASCMs). SSDS Mk 2 is designed to expedite that process. Consisting of software and commercial-off-the-shelf (COTS) hardware, SSDS Mk 2 integrates radar systems with anti-air weapons, both hardkill (missile systems) and softkill (decoys).

SSDS Mk 2 includes embedded doctrine to provide an integrated detect-through-engage capability with options ranging from use as a tactical decision aid to use as an automatic weapons system to respond with hardkill and softkill systems.

Although SSDS Mk 2 will not improve the capability of individual sensors, it enhances target tracking by integrating the inputs from several different sensors to form a composite track. For example, SSDS Mk 2 will correlate target detections from individual radars, the electronic support measures (ESM) system (radar warning receiver), and the identification-friend or foe (IFF) system, combining these to build composite tracks on targets while identifying and prioritizing threats. Similarly, SSDS Mk 2 will not improve the capability of individual weapons, but should expedite the assignment of weapons for threat engagement, and provide a "recommend engage" display for operators, or if in automatic mode, initiate weapons firing, ECM transmission, chaff or decoy deployment, or some combination of these.

SSDS Mk 2 integrates previously "stand-alone" sensor and engagement systems for aircraft carriers and amphibious warfare ships, thereby supporting the Joint Vision 2010 concept of full-dimensional protection, by providing a final layer of self-protection against air threat "leakers" for individual ships. By ensuring such protection, SSDS Mk 2 contributes indirectly to the operational concept of precision engagement, in that strike operations against targets are executed from several of the platforms receiving SSDS Mk 2.

Ada in Poland

From: Adrian Hoe <abyhoe@gmail.com>
Newsgroups: comp.lang.ada
Subject: Re: Current status of Ada?
Date: Tue, 18 Sep 2007 08:26:14 −0000

> What kind of military software is developed in Ada in Poland?

They are into both military and civil. Michal Nowak (you can google his name in C.L.A.) is a close friend of mine and he is working in one of the company now. Real safety critical stuff involving ARINC! And two company in Poznan doing some contract work for two aerospace industries in USA!

Indirect Information on Ada Usage

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

Job Description: Maryland

[..] 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. This means that they are putting a lot of money into helping us grow our business. Anyone looking to make a change to a position with a lot of growth potential should consider this opportunity. If you don't meet the minimum requirements below, you will not be considered.

Project:

The Radar Digital Signal Injection System (RDSIS) is a 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 [...]

Job Description: Germany

[...] we are looking for a Software Engineer.

Requirements:

- Software update according to programming rules
- Software Detailed Design Document generation based on Prototype Code
- Performance of design and code reviews to check compliance with standards
- Hardware/Software Integration
- Hardware/Software Integration Testing
- Software Acceptance Testing

Required skills:

- Programming language Ada 83 or 95
- Generation of software detailed design documentation
- Software development according to RTCA/DO-178B
- Motorola Controller MPC565
- Lauterbach BDM and/or NEXUC debugger

Recommended skills:

- Greenhills MULTI Ada
- DOORS
- DIMENSIONS
- LDRA Testbed

Start: asap

Duration: 6 months with possible extension

Location: Munich Germany

Language: English

Ada in Context

ARM in info format

Newsgroups: comp.lang.ada
Subject: Ada reference manual source processing program
From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Fri, 28 Sep 2007 03:22:04 −0400

I've finally gotten around to posting the source for the Ada code that produces the Ada Reference Manual, in several output formats (including Emacs info), on my web page:

http://stephe-leake.org/ada/arm.html

I fixed a bug in the info version in September; the current version of the info manuals is arm2005-20070928.tar.gz

At Randy Brukardt's request, the source zip does not include the actual ARM sources; you can get those directly from the official ARM website http://www.ada-auth.org/arm.html

ARM in LaTeX

From: Randy Brukardt <randy@rrsoftware.com>
Newsgroups: comp.lang.ada
Subject: Re: ARM and AARM pdf's: no bookmarks
Date: Wed, 14 Nov 2007 23:25:55 −0600

> When using ARM and AARM pdf's from AdaIC site, they show no bookmarks in Adobe Reader. Many modern books come with bookmarks for chapters and subchapters, which are shown hierarchically in PDF viewers, allowing for quick jump to a section of interest. Who should be contacted to suggest the official pdf's are processed to get such bookmarks?

You'd probably want to ask me.

But I have absolutely no idea of how to add bookmarks to the PDF in an automated way, given that the original files are processed through Microsoft Word. (Adding them by hand surely won't be practical.) The existing table-of-contents information doesn't seem to make bookmarks when the document is printed to Distiller. I've tried some of the other conversion plugins, but they don't seem to be able to use the proper fonts

(which makes the Unicode examples unreadable). We don't have the budget to spend time figuring this out (we have no requirement to make PDFs for anyone).

In any case, we produced the PDF versions solely for the purpose of printing the standard. If you want a version with links, use the HTML version (which also gives you access to a decent search facility and a fully linked syntax).

If you really, really wanted to upgrade the PDFs, I'd suggest downloading the Standard's construction tools and doing this yourself. I'd be happy to give you some pointers if you wanted to do that (and the handful of hand-corrections that are needed to make the document printable).

Randy Brukardt, ARG editor; Editor, ISO/IEC 8652

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Thu, 15 Nov 2007 04:41:13 −0500
Subject: Re: ARM and AARM pdf's: no bookmarks
Newsgroups: comp.lang.ada

> But I have absolutely no idea of how to add bookmarks to the PDF in an automated way, given that the original files are processed through Microsoft Word. (Adding them by hand surely won't be practical.)

That's partly why I started a LaTeX version of the ARM; LaTeX to PDF can produce bookmarks, as well as all the other hyperlinks.

But it's a big job.

See http://stephe-leake.org/ada/arm.html for the version of the construction tools with a start on LaTeX output.

Localization and Ada

From: Vadim Godunko <vgodunko@gmail.com>
Date: Fri, 19 Oct 2007 12:23:45 −0000
Subject: Re: Ada and locale
Newsgroups: comp.lang.ada

> I would like to know how I can benefit from the locale support in my operating system. By locale I mean the set of properties of the I/O system that allow to customize formatting of numbers, collating, character classification and such.

ARM don't define any way for application localization. You must use some external library.

GUI toolkits usually include such support. I don't known about GTK, but Qt have QLocale class for numeric formatting customization and support characters and strings operations through QString and QChar classes.

From: Jacob Sparre Andersen <sparre@nbi.dk>
Date: Fri, 19 Oct 2007 23:06:55 +0200
Subject: Re: Ada and locale
Newsgroups: comp.lang.ada

For example using the package GtkAda.Intl (which only covers basic "gettext" stuff).

Allocators and exceptions

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 10 Sep 2007 21:36:56 −0500
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

> What happens when during the initialization of the newly allocated object an exception is raised?

> I cannot find anything in the AARM that covers this case. What I want to find exactly is the *guarantee* that the allocated memory is automatically reclaimed. Any relevant paragraph numbers are highly welcome.

(1) Nothing in the Ada standard is about "goodness". In particular, there is nothing anywhere in the standard that resources like memory ever get reclaimed. I suspect most implementers will in fact do reclamation (and avoid leaks), but it is not part of the Ada language as described by the standard.

(2) I believe that the current wording of the standard *requires* that reclamation *not* be performed in examples like this, at least if there are any controlled components in the type. That's because there is no permission in Ada to do finalization early — it has to be done only if the object is explicitly destroyed or when the master goes out of scope — which for an allocated object is when the *type* goes out of scope.

(Not everyone agrees with the above opinion, but everyone does agree that it is an issue in some cases. But there is nothing close to an agreement on how to fix the standard, so don't hold your breath waiting for a fix...)

Yes, this also means that an Ada compiler implementing garbage collection is mostly likely incorrect. It's highly unlikely, however, that anyone will be testing for such "errors" formally. I did write an ACATS-style test for a case like this and determined that most compilers do in fact finalization the object at the appropriate time: which suggests that they leak memory in this case.

Moral: Never, ever, write code that intentionally raises an exception during an allocator. (Unintentional exceptions are just plain bugs and ought to get fixed in testing.) Better still, don't use any allocators at all (use the predefined containers if you need dynamic memory management).

From: Maciej Sobczak <maciej@msobczak.com>
Date: Mon, 10 Sep 2007 14:48:56 −0700
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

> [...] Exceptions in constructors is a bad idea.

No, it's a very good idea. Otherwise you have to deal with half-baked objects, which is Even Bigger Mess (tm). [...]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 11 Sep 2007 11:16:31 +0200
Organization: cbb software GmbH
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

This is what you get when the exception is propagated out of a constructor. It breaks the abstraction, necessarily. You cannot handle this unless you accept the idea that one can always view an object as an aggregate of other objects. This in turn would imply 1) types matched by structure, 2) broken encapsulation. You want the compiler to invent partial constructors/destructors, it is a difficult problem, probably undecidable.

From: Maciej Sobczak <maciej@msobczak.com>
Date: Wed, 12 Sep 2007 05:36:54 −0700
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

> C++ doesn't seem to handle deallocation etc. of sibling components either, so I'm not sure I understand.

It does. In case of exception the already constructed components are rolled back. This works for components of array as well. [...]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 12 Sep 2007 17:19:01 −0500
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

Ada does of course finalize any components that have already been constructed. It doesn't just drop them on the floor!! The problem (if you can call it that) with Ada is that is clearly defines when that will happen. You want it to happen *earlier* than that definition.

One could argue that that definition is wrong, but it is what it is.

For what it's worth, the definition you seem to want would be extremely expensive to implement in Janus/Ada: every allocated component would need a dedicated (compiler generated) exception handler in order to be able free the associated memory immediately. The effect would be to make allocators 10 times bigger and possibly 10 times slower. (It surely would be that much for our [obsolete] MS-DOS compilers, which used a heap of our own design; I'm not sure how expensive the Windows heap allocations are so it might be somewhat less.) Programs that do a lot of allocation could have a pretty significant

performance impact (and that would include the containers libraries).

From: Simon Wright <simon.j.wright@mac.com>
Date: Wed, 12 Sep 2007 21:53:24 +0100
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

[...] It certainly seems a bad idea to allow a Constraint_Error to propagate unhandled. But what would be wrong with dealing with the problem and then raising an appropriate exception from the constructor? (even Constraint_Error if gnat makes sense).

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 12 Sep 2007 17:32:52 −0500
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

Because Ada *requires* storage leakage in such cases (although some compilers ignore the language definition and finalize anyway). An allocated object cannot be finalized until it's *type* is finalized or until an Unchecked_Deallocation is called — and an Unchecked_Deallocation is not going to be called if a constructor propagates an exception. So the (inaccessible) object is supposed to hang around for a long, long time.

The "proper" way to handle this is to ensure that default initialize of an object never propagates an exception, and then wrap the allocator properly:

```
type Access_T is access all T;
procedure Free is new
Unchecked_Deallocation (T, Access_T);
function Alloc_Object (...) return
                    Access_T is
  A_T : Access_T := new T; -- Default
                           -- initialized.
 begin
     A_T.all := <constructor>;
     return A_T;
 exception
     when others => Free(A_T);
       return null;
 end Alloc_Object;
```

But I'm not going to argue that this is an ugly and complex way of handling this.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 12 Sep 2007 22:36:05 −0500
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

For what it's worth, the rule in the RM is 7.6.1(10), which says:

"Immediately before an instance of Unchecked_Deallocation reclaims the storage of an object, the object is finalized. If an instance of Unchecked_Deallocation is never applied to an object created by an allocator, the object will still exist when the corresponding master completes, and it will be finalized then."

From: Maciej Sobczak <maciej@msobczak.com>
Date: Thu, 13 Sep 2007 02:43:41 −0700
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

Yes, this is a very reasonable approach. I would even propagate (or translate) the exception out instead of returning null — this can make it more plausible to work with at the call site.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 12 Sep 2007 22:42:27 −0500
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

> I guess I had misunderstood what's meant by 'constructor', because this is just what I had in mind ... and there are far worse things to leak than memory, such as locks, file handles, data structure integrity etc, and those we can handle even in a constructor (i. e. function returning a value of the type rather than a pointer to a new value of the type).

The problem with this sort of construction (besides that it is clunky) is that it doesn't work for limited types without breaking abstraction. OTOH, the leak in this case doesn't bother me too much, because constructor failure ought to be rare and it is also rare to be creating a lot of objects — so it usually doesn't matter. Moreover, safety critical applications aren't going to be using allocators in the first place, and very long-running applications are likely to have problems with memory fragmentation even if they don't leak any memory — unless they have a lot more memory available than they're going to need. Still, the leak is uncomfortable — it doesn't match Ada's goals.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Wed, 12 Sep 2007 08:29:08 −0400
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

> In this case I want the constructor to be rolled back. Without exceptions (and rollback) the only option for handling errors in initialization of (sub)components is to leave them half-baked.

The point is that the constructor itself must do the roll-back, and leave the object in a consistent state.

The rule should be:

Constructors should not propagate exceptions up; they must handle all exceptions internally.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 12 Sep 2007 17:25:07 −0500
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

> What should the constructor do *after* handling the exception? Leave the object half-baked?

No, unbaked.

> Do you want to introduce additional states to the object design just to handle the "oops I'm not initialized" case?

No, because you must have those states anyway. Ada allows an object to be finalized multiple times (either by an explicit call to Finalize, or in some obscure cases involving aborts), so you have to have an invalid (not between initialization and finalization) state in any Ada controlled object. (Blindly doubly finalizing an object is likely to be a serious bug, because of calling Unchecked_Deallocation twice on the same object or similar gaffes.) Any controlled type that doesn't have an "I'm not valid" state is wrong, period.

Once you have such a state, having an uninitialized object is not a disaster.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 12 Sep 2007 17:45:07 −0500
Subject: Re: Allocators and exceptions
Newsgroups: comp.lang.ada

> How to reconcile this with:

> 1. objects allocated on the stack

> 2. all sorts of temporal objects the compiler is allowed to create (and thus allocate somehow, somewhere)

> 3. a permission given to collect garbage?

That's the *real* problem: Ada has no such permission when it comes to objects with non-trivial finalization. It is defined precisely where they are going to be finalized, and there is no permission to do it early without an explicit call (which doesn't exist).

As it stands, a compiler that does the right thing (in the sense of avoiding a memory leak when it is certain that no reference to the object remain) is actually wrong vis-a-vis the language definition.

> Anyway, presuming that the constructor shall clean its mess before propagating any exceptions, there is no any object here to "deallocate" (I would say "destroy").

Surely the top-level object's memory was allocated, and there is no place in the language that would ever require Deallocate to be called. Moreover, even a friendly compiler could not do that without violating the language definition.

> A related issue, why on earth "new" is allowed to propagate anything but Storage_Error or else Program_Error?

"new" doesn't propagate anything other than Storage_Error or Program_Error. The initialization expression, OTOH, can propagate anything it wants. Those are separate things from a language perspective; the problem is that you can't *write* them separately.

From: Maciej Sobczak <maciej@msobczak.com>

Date: Mon, 10 Sep 2007 05:16:07 −0700
Subject: Re: Allocators and exceptions => Read Me First
Newsgroups: comp.lang.ada

> pragma Restrictions (Immediate_Reclamation) ;

It does not apply to objects created by the allocator.

It is useful only for those objects that are created implicitly, for example return values of unconstrained types. [...]

Emmett Paige's 1997 DoD Memo

From: Richard Riehle <adaworks@sbcglobal.net>
Newsgroups: comp.lang.ada
Subject: Re: History of Ada — and about the NYU DOS version
Date: Fri, 02 Nov 2007 13:51:48 GMT

> The DOD decided to use more cheaper versions of computer languages, such as C.

The DoD did not decide to use cheaper languages such as C. This incorrect assessment of Mr. Paige's (then Assistant Secretary of Defense) memo lifting the Ada mandate has been widely disseminated. Rather, Mr. Paige opened the door to the use of other languages so Ada would compete on its merits instead of on a strict policy level.

In Mr. Paige's memo, he even cited Ada's success along with his belief that, since Ada had proven to be a valuable tool for DoD software, it was now able to stand on its own in the competitive environment of programming language choice.

Mr. Paige expressed the hope that Ada would continue to be used for vital DoD software.

Many in the DoD and elsewhere misinterpreted Mr. Paige's memo lifting the Ada mandate. Unfortunately, this misinterpretation is now so widespread that many DoD personnel are of the opinion that Ada has been "forbidden" for military software. Somehow, the simple lifting of the mandate has gone through a series of stages: Ada is no longer required; Ada is no longer supported (closing of the AJPO); Ada is no longer to be used; Ada is now forbidden.

The reality is that Mr. Paige, and his original DoD memo, foresaw Ada as continuing to serve the needs of military software far into the future, but more as one of a set of options than as the sole [mandated] option.

Ada continues to be used for DoD software systems, though not as widely as it once was, primarily due to the misinterpretation of Mr. Paige's memo.

[See also same topic in AUJ 28-3 (Sep 2007), pp.159–161 —su]

TIOBE Programming Community Index

From: Martin Krischik <krischik@users.sourceforge.net>
Subject: TIOBE Programming Community Index for November 2007
Newsgroups: comp.lang.ada
Date: Sun, 04 Nov 2007 18:56:29 +0100

[..] I don't like the index, but it is there, widely used and I think we have to deal with it.

And currently we deal badly — place 20 — one down and we are off the scale:

http://www.tiobe.com/tpci.htm

[...] But then it shows that they have a dedicated comunity which can fix something. Why can't we? [...]

From: Martin Krischik <krischik@users.sourceforge.net>
Newsgroups: comp.lang.ada
Subject: Re: TIOBE Programming Community Index for November 2007
Date: Mon, 05 Nov 2007 08:27:03 +0100

[...] it they just count hits. If you got to google [...] you get 220.000 pages. Note the use of +"Ada programming" — without the + and the " it would be a lot more. I think adding:

<META NAME="KEYWORDS" CONTENT="Ada programming">

to every Ada related page would do the trick.

From: Martin Krischik <krischik@users.sourceforge.net>
Newsgroups: comp.lang.ada
Subject: Re: TIOBE Programming Community Index for November 2007
Date: Mon, 05 Nov 2007 10:35:26 +0100

> Just "Ada" isn't enough? I think I have no single "Ada programming" on my pages.

The way they described it you will need the exact string "Ada programming" with one space in between. The main reason why we chosen "Ada programming" when we renamed the wikibook. Or created a category "Category:Ada programming" on Wikipedia. Or have you noticed the "Ada programming, © 2005,2006 the Authors, Content is available under GNU Free Documentation License." on all the other Wiki pages.

And for a short time it worked moving Ada up 2 .. 3 places. But not in the long run — you need to keep momentum. But I already tweaked most of the pages I have access to.

BTW: there is a reason for it. Just searching Ada programming on YouTube gives you lots of hits on women forenames and/or TV programs.

From: Manuel Gómez <mgrojo@gmail.com>
Date: Mon, 05 Nov 2007 11:46:02 −0800
Subject: Re: TIOBE Programming Community Index for November 2007
Newsgroups: comp.lang.ada

They say:

"From this month on, we have stopped monitoring Google groups because it is not representative anymore. Instead we have added YouTube for a small percentage. The choice for YouTube might seem strange but it is now #4 on the Alexa.com chart and people tend to upload lectures and "how to" videos on this site. The top 3 programming languages on YouTube is Java, C++, and (surprisingly) Python."

Given that they have added YouTube we should think on uploading some videos to that site. Some old ones can be found in [http://www.adapower.com/...]

AdaCore has a great repository of Ada videos but I wonder whether they would like that people upload their videos to YouTube.

Protected Objects in Ravenscar

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Organization: cbb software GmbH
Date: Wed, 5 Sep 2007 09:46:39 +0200
Subject: Re: Ravenscar-compliant bounded buffer
Newsgroups: comp.lang.ada

> While I don't know much about the Ravenscar profile (other than what it is) I am familiar with the use of protected types.

> The sample code is a good illustration of "abstraction inversion". [...] The code uses a high level abstraction "protected type" to create a low level abstraction "Binary_Semaphore". The code uses two semaphores to restrict access to the bounded buffer. In this simple example it is easy to follow, but in a more complex example pairing Acquire's and Release's can be a chore.

That is because of the Ravenscar limitation of one entry per protected object. The solution Maciej presented is based on splitting one protected object of two entries into two, each controlling access to its end of FIFO. Protected objects don't compose so the result. [...]

From: Robert A Duff <duff@adacore.com>
Newsgroups: comp.lang.ada
Subject: Re: Ravenscar-compliant bounded buffer
Date: Thu, 06 Sep 2007 22:36:59 −0400

[...] simpler run-time system means easier to verify that it does what's intended. I suppose that's good for users of it, unless they need re-implement all of Ada "by hand" on top of the supposedly simpler run-time system.

> [...] the point about abstraction inversion stands.

Agreed. As I said, it's a choice. If you really need to put multiple tasks on entry queues, then you probably don't want Ravenscar. If you can easily live with the limitations of Ravenscar, you might benefit from the simplicity.

From: Robert A Duff <duff@adacore.com>
Newsgroups: comp.lang.ada
Subject: Re: Ravenscar-compliant bounded buffer
Date: Thu, 06 Sep 2007 10:06:51 −0400

> Ada 83 was restrictive in ways that were found to be overly restrictive for practical application. Some of these restrictions were relaxed with Ada 95. Perhaps the next round of Ravenscar will do the same.

I don't see any need to relax Ravenscar, because if you want to use features not allowed by Ravenscar, you don't have to restrict yourself to Ravenscar. It's a free choice. I suppose we could argue about whether the exact set of restrictions is appropriate, but the whole point is to be restrictive, so the run-time system can be simplified (as compared to a run-time system that supports full Ada).

From: Jean-Pierre Rosen <rosen@adalog.fr>
Newsgroups: comp.lang.ada
Subject: Re: Ravenscar-compliant bounded buffer
Date: Wed, 05 Sep 2007 10:30:09 +0200
Organization: Adalog

[...]

The limitation on only one task per queue is intended to guarantee bounded waiting time. Any solution that tries to work around the Ravenscar rules in order to have longer queues will violate this restriction at some point, or have to manage explicit lists — thus violating the spirit of the profile!

[See also "Why is task termination disallowed in Ravenscar?" in AUJ 28-1 (Mar 2007), pp.31–32. —su]

Allocation of large objects

From: Pascal Obry <pascal@obry.net>
Date: Tue, 30 Oct 2007 08:28:37 +0100
Subject: Re: Largest size array in Gnat 2005 for the PC?
Newsgroups: comp.lang.ada

> What is the largest array (in storage units) that you can declare in Gnat 2005 for the PC?

> Does pragma Storage_ size affect this and if so where would you place it in a procedure?

It depends if you want to allocate it on the stack or on the heap. The stack is often smaller but the size can be changed at link time. The heap can use all the memory (physical + virtual) that you have on your computer. There is a limit in the size allocated by a single object imposed by the OS depending on the architecture (32bits / 64bits).

From: Stefan Bellon <sbellon@sbellon.de>
Date: Tue, 30 Oct 2007 16:00:21 +0100
Organization: Comp.Center (RUS), U of Stuttgart, FRG
Subject: Re: Largest size array in Gnat 2005 for the PC?
Newsgroups: comp.lang.ada

> > I am using the stack. how do I change this at link time using GPL?

> There is no option for that use "new" to allocate the array instead of declaring it on the stack.

While I agree with the given suggestion to allocate large objects on the heap instead of the stack, there is a way to increase the stack size on Windows. We use something like the following in our projects where we need a larger stack on Windows:

```
package Linker is
   case OS is
      when "Unix" =>
         null;
      when "Windows_NT" =>
         for Default_Switches ("ada") use
          ("--stack=0x2000000,0x10000");
   end case;
end Linker;
```

From: Stefan Bellon <sbellon@sbellon.de>
Date: Wed, 31 Oct 2007 10:22:13 +0100
Organization: Comp.Center (RUS), U of Stuttgart, FRG
Subject: Re: Largest size array in Gnat 2005 for the PC?
Newsgroups: comp.lang.ada

> I tried the linker switch -Wl,--stack=0x10000000 in GPL but what is the second number "0x10000" for?

The first is the stack reserve and the second is the stack commit size. Specifying the commit size is optional. Default are 2 MB/4 KB respectively, the above values are factor 16 to the default.

From: Adam Beneschan <adam@irvine.com>
Date: Tue, 30 Oct 2007 12:06:39 −0700
Subject: Re: Largest size array in Gnat 2005 for the PC?
Newsgroups: comp.lang.ada

> [...] if your CPU and operating system is limited to 32 bits then GNAT defines:

> System.Memory_Size : constant := (2 ** 32) ;

I should point out that you should *not* use System.Memory_Size for this purpose unless you're using GNAT and are absolutely certain your code will not be compiled with another compiler. The original definition of Memory_Size had to do with the amount of available memory, not the amount of memory that could be accessed with an address (whether the memory existed or not); starting with Ada 95, the AARM has said:

It is unspecified whether this refers to the size of the address space, the amount of physical memory on the machine, or perhaps some other interpretation of "memory size." In any case, the value has to be given by a static expression, even though the amount of memory on many modern machines is a dynamic quantity in several ways. Thus, Memory_Size is not very useful. [13.7(33.a)]

From: Stefan Bellon <bellon@software-erosion.org>
Date: Thu, 1 Nov 2007 09:44:20 +0100
Subject: Re: Largest size array in Gnat 2005 for the PC?
Newsgroups: comp.lang.ada

> If GNAT placed "IMAGE_FILE_LARGE_ADDRESS_AWARE" in the process headers for GNAT executables then you could use up to 3GB of memory.

> Seems like a simple thing to implement.

```
package Linker is
   case OS is
      when "Unix" =>
         null;
      when "Windows_NT" =>
         for Linker_Options use
         ("-Wl,--large-address-aware");
   end case;
end Linker;
```

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

January 07-09 — 9th **International Conference on Verification, Model Checking, and Abstract Interpretation** (VMCAI'2008), San Francisco, California, USA. Co-located with POPL'2008. Topics include: program verification, program certification, model checking, static analysis, type systems, etc.

☺ January 10-12 — 35th Annual ACM **SIGPLAN-SIGACT Symposium on Principles of Programming Languages** (POPL'2008), San Francisco, California, USA. Topics include: fundamental principles and important innovations in the design, definition, analysis, transformation, implementation and verification of programming languages, programming systems, and programming abstractions.

January 07-08 — ACM **SIGPLAN Workshop on Partial Evaluation and Program Manipulation** (PEPM'2008). Topics include: program analysis, program generation and program transformation.

January 13 — 2008 **International Workshop on Foundations of Object-Oriented Languages** (FOOL'2008), San Francisco, California, USA. Topics include: language semantics, type systems, program analysis and verification, concurrent and distributed languages, language-based security issues, etc.

January 16 — 2nd **International Workshop on Variability Modelling of Software-intensive Systems** (VaMoS'2008), Essen, Germany.

☺ January 24 — **Software-Workshop - Effiziente Entwicklung zuverlässiger Software und methodisches Instrumentarium**, Karlsruhe, Germany. Organized among others by Ada-Deutschland and Fachgruppe Ada of the Gesellschaft für Informatik (GI). Includes: talks on "Ada 2005 for real-time, embedded and high-integrity systems" by José Ruiz, AdaCore France, and "Hibachi -the Eclipse Ada Development Toolset" by Tom Grosman, Aonix.

February 12-14 — IASTED **International Conference on Parallel and Distributed Computing and Networks** (PDCN'2008), Innsbruck, Austria. Topics include: Parallel Programming, Parallel Processing, Reusability, Reliability, Scheduling, Modelling and Simulation, Distributed Real-Time Systems, Compilers, Fault Tolerance, Performance Evaluation, Real-Time and Embedded Systems, Applications, etc.

February 12-14 — 5th IASTED **International Conference on Software Engineering** (SE'2008), Innsbruck, Austria. Topics include: Software Design and Development, Software Tools, Software Maintenance, Software Metrics and Testing, Reliability, Quality Assurance, Software Evaluation, Reusability, Verification and Validation, Fault Tolerance, Object-Oriented Analysis and Design, Security, Software for Parallel and Distributed Systems, Education, Model-Driven Development, etc.

February 13-15 — 16th Euromicro **International Conference on Parallel, Distributed and Network-based Processing** (PDP'2008), Toulouse, France. Topics include: Parallel Computer Systems (embedded parallel and distributed systems, fault-tolerance, ...); Models and Tools for Parallel Programming Environments; Advanced Applications (numerical applications with multi-level parallelism, real time distributed applications, ...); Languages, Compilers and Runtime Support Systems (object-oriented languages, dependability issues, scheduling, compilers for multicore architecture, ...); etc.

February 18-21 — 7th IEEE/IFIP **Working Conference on Software Architecture** (WICSA'2008), Vancouver, BC, Canada. Topics include: Software Architecture Modeling and Analysis Methods and Tools; Architecture Description Languages and Model Driven Architecture; Software Architecture for Legacy Systems and Systems Integration; Education, and Certification of Software Architects; Industrial case studies; etc.

February 25-26 3rd **International Workshop on Systems Software Verification** (SSV'2008), Sydney, Australia. Theme: "Real Software, Real Problems, Real Solutions". Topics include: static analysis, model-driven development, embedded systems development, programming languages, verifying compilers, software certification, software tools, experience reports, etc. Deadline for registration: January 11, 2008.

February 25-29 7th **International Conference on Composition Based Software Systems** (ICCBSS'2008), Madrid, Spain. Theme: "Weaving Composite Systems". Topics include: composibility and integration scenarios, open source ecosystems, species of legacy systems, technologies for interoperability, standards, legal issues (including FOSS), etc.

March 02-07 14th **Conference on Languages and Models with Objects** (LMO'2008), Montréal, Québec, Canada. Topics include (in French): Programmation par objets: langages, interprétation, compilation; objets et types; environnements de programmation; ... Composants, Services et Objets distribués: modèles; intergiciels; raisonnement compositionnel; parallélisme; interopérabilité; ... Génie des objets et des modèles: cycle de vie des objets et des modèles; évolution, rétro-conception, réutilisation, versions; sûreté de fonctionnement, spécifications formelles; processus de développement; hiérarchies, frameworks, patterns; ertc. Applications: objets et algorithmique; objets métier; objets pour les IHM, les télécommunications, les systèmes embarqués, le multimédia, la chimie, etc.

☺ March 04-07 CISIS2008 - **International Workshop on Multi-Core Computing Systems** (MuCoCoS'2008), Barcelona, Spain. Topics include: programming languages and models; performance modeling and evaluation of multi-core systems; tool-support for multi-core systems; compilers, runtime and operating systems; etc.

☺ March 12-14 SIAM **Conference on Parallel Processing for Scientific Computing** (PP'2008), Atlanta, Georgia, USA. Topics include: Programming languages, models, and compilation techniques; The transition to ubiquitous multicore/manycore processors; Tools for software development and performance evaluation; Parallel computing in industry; Distributed/grid computing; Fault tolerance; etc.

☺ March 12-15 39th ACM **Technical Symposium on Computer Science Education** (SIGCSE'2008), Portland, Oregon, USA. Visit the ACM SIGAda booth!

March 16-20 23rd ACM **Symposium on Applied Computing** (SAC'2008), Fortaleza, Ceara, Brasil.

- ☺ Mar 16-20 *Track on* **Object-Oriented Programming Languages and Systems** (OOPS'2008). Topics include: Design and implementation of novel abstractions, constructs and mechanisms; Multi-paradigm features; Language features in support of adaptability; Component-based programming; Generative programming; Program structuring, modularity; Distributed objects and concurrency; Middleware; Compilation techniques; etc.
- Mar 16-20 *Technical Track on* **Software Verification.** Topics include: Data flow analysis, control flow analysis, type effect systems, constraint systems and abstract interpretation techniques for verification; Techniques to validate system software (such as compilers) as well as assembly code or bytecode; Software certification and proof carrying code; Integration of formal verification into software development projects; etc.
- Mar 16-20 *Track on* **Software Engineering** (SE'2008). Topics include: Component-Based Development and Reuse; Dependability and Reliability; Fault Tolerance and Availability; Maintenance and Reverse Engineering; Verification, Validation, Testing, and Analysis; Formal Methods and Theories; Empirical Studies, Benchmarking, and Industrial Best Practices; Applications and Tools; Distributed, Embedded, Real-Time, High Performance, Highly Dependable Systems; etc.

Mar 29 – Apr 06 **European Joint Conferences on Theory and Practice of Software** (ETAPS'2008), Budapest, Hungary.

- Mar 29–Apr 6 11th **International Conference on Fundamental Approaches to Software Engineering** (FASE'2008). Topics include: SE as an engineering discipline, Specification and design, Software evolution, Validation and verification, etc.
- April 05 8th **Workshop on Language Descriptions, Tools and Applications** (LDTA'2008). Topics include: Program analysis, transformation, generation and verification; Reverse engineering and reengineering; Refactoring and other source-to-source transformations;

Language definition and language prototyping; Debugging, profiling and testing; IDE construction; Compiler construction; etc.

Mar 31 – Apr 04 7th **International Conference on Aspect-Oriented Software Development** (AOSD'2008), Brussels, Belgium.

Mar 31 – Apr 04 15th **Annual IEEE International Conference and Workshops on the Engineering of Computer Based Systems** (ECBS'2008). Belfast, Northern Ireland. Topics include: Component-Based System Design; Design Evolution; Distributed Systems Design; ECBS Infrastructure (Tools, Environments); Education & Training; Embedded Real-Time Software Systems; Integration Engineering; Model-Based System Development; Modelling and Analysis of Complex Systems; Open Systems; Reengineering & Reuse; Reliability, Safety, Dependability, Security; Standards; Verification & Validation; etc. Deadline for early registration: February 26, 2008.

☺ April 01-04 3rd **European Conference on Computer Systems** (EuroSys'2008), Glasgow, UK. Topics include: All areas of operating systems and distributed systems; Systems aspects of: Dependable computing, Parallel and concurrent computing, Distributed algorithms, Programming language support, Real-time and embedded computing, Security, ...; Experience with existing systems; Reproduction or refutation of previous results; Negative results; Early ideas.

April 01-04 12th **European Conference on Software Maintenance and Reengineering** (CSMR'2008), Athens, Greece. Theme: "Developing Evolvable Systems". Topics include: Software migration strategies and technologies; Empirical studies in maintenance and reengineering; Experience reports on evolution, maintenance and reengineering; Education in maintenance and reengineering; etc.

April 01-04 6th ACS/IEEE **International Conference on Computer Systems and Applications** (AICCSA'2008), Doha, Qatar. Topics include: Parallel programming models, Programming environments and tools, Parallelizing compilers, Distributed systems, Parallel embedded systems, Formal Methods for Security, Software Design and Development, Model-Driven Development, Fault Tolerant Software Systems, Formal Methods, Verification, Validation, etc.

April 09-11 2nd **International Conference on Tests And Proofs** (TAP'2008), Prato (near Florence), Italy.

☺ 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.

Apr 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.

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. Deadline for submissions: January 10, 2008.

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. Deadline for submissions: January 8, 2008 (abstracts), January 15, 2008 (papers).

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. Deadline for submissions: January 8, 2008 (abstracts), January 15, 2008 (papers).

☺ 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. Deadline for submissions: January 7, 2008 (papers, tutorials). Deadline for early registration: March 15, 2008.

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: January 2, 2008 (workshops), January 28, 2008 (abstracts), February 5, 2008 (papers), April 7, 2008 (tools and posters). Deadline for early registration: March 15, 2008.

♦ 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 (approval pending). Deadline for submissions: January 13, 2008 (industrial presentations).

June 17-19 — 2nd IEEE & IFIP **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. Deadline for submissions: January 21, 2008 (abstract), January 28, 2008 (papers).

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. Deadline for paper submissions: January 18, 2008.

☺ 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. Deadline for submissions: March 7, 2008.

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. Deadline for technical paper submissions: February 8, 2008.

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. Deadline for submissions: February 10, 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. Deadline for submissions: February 4, 2008 (draft papers).

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

July 07-13 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, Applications and case studies, Verification in industrial practice, etc. Deadline for submissions: January 28, 2008 (papers, CAV Award nominations).

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. Deadline for submissions: January 14, 2008 (abstracts), January 21, 2008 (full papers).

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. Deadline for submissions: January 12, 2008 (abstracts), January 19, 2008 (full papers).

☺ 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. Deadline for submissions: January 25, 2008 (papers, workshops).

☺ 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: March 15, 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. Deadline for paper submissions: February 4, 2008.

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.

♦ Oct 26-30 2008 ACM **SIGAda Annual International Conference** (SIGAda'2008), Portland, Oregon, USA. Sponsored by ACM SIGAda (approval pending). Topics include: Safety, security and high integrity development issues; Language selection for a high reliability system; Use of ASIS for new Ada tool development; Mixed-language development; High reliability software engineering education; High reliability development experience reports; Static and dynamic code analysis; Use of new Ada 2005 features/capabilities; etc. Deadline for submissions: May 10, 2008 (technical articles, extended abstracts, experience reports, workshops, panel sessions, and tutorials).

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

13th International Conference on Reliable Software Technologies

Ada-Europe 2008

Venice, Italy
16-20 June 2008

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

Highlights

- First-class keynote speakers
  - Enjoyable, contentious, visionary
- A rich technical program
  - Currently in the making
- An outstanding social program
  - Including one for accompanying persons
- A very special venue
  - A former monastery dating back to the 14th century in the heart of Venice

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

Secure software-download as part of a complex business process

Igor Furgel, Lars Hanke

T-Systems GEI GmbH, Rabinstraße 8, D-53111 Bonn; E-Mail: Lars.Hanke@T-Systems.com

Abstract

As embedded systems become more powerful, concepts known from the personal computer market are ported to reliable systems. Updatable multi-application systems can be technically realized and can enable new business fields. This article shows that the requirements concerning upgrade security imposed by reliable systems may be entirely different from the PC market. The influence on the design of an upgrade business process, technical and organizational infrastructure, and on the total cost are analyzed.

Keywords: Software Update, Security, Operating System, Security Domains, Business Process, Risk Analysis

1 Introduction

Embedded systems have penetrated all aspects of critical control systems such as vehicle and airplane electronics, industrial automation and health-care systems. The option to easily repair devices, enable add-on functionality or integrate additional services gives rise to marketing requests for software updates of such systems in the field. Usually, the main focus is set on technical issues concerning specific protocols. In this paper we focus on the business case related to updates of reliable software and in particular on the required infrastructure for its implementation, which is always assumed in the technical papers. We show that most of the business process must be defined before development begins and that it rigidly defines organizational and technical infrastructure as well as development overhead. From a commercial point of view the decision whether or not software updates shall be at all supported may be more crucial than how technical details shall be implemented.

There are two main reasons for software changes. First, the original software may contain errors, which influence the functionality of the device, or even the safety of the system's users. Repairing such faults by a software update offers considerable cost-benefits. Techniques like online updates may even render certain product recalls unnecessary.

Second, software updates enable an entire market of value added services. Using the same hardware for different models or even entirely different control-units, and configure these by software for the specific task and model, can greatly reduce the total hardware development cost.

Software updates for virtually all of these purposes currently have a long tradition in the environment of personal computers (PC). The PC market differs in at least three vital aspects from the reliable systems' market.

1. Customers in the PC market are mostly able to establish online connections to the software developer at any time, and they are expected to perform the update or maintenance procedure by themselves.
2. The hardware platform used in the PC market is a multi-purpose platform. The configuration of a PC is generally unknown, and different vendors compete for market shares for the same type of software.
3. Failure in PC systems rarely induces any liability exceeding the software price. In particular, PC systems are never considered as relevant to life safety.

These aspects give rise to the key issues when devising a business process for updating reliable embedded systems:

1. How is the update data transported to the system?
2. Who will perform the update installation?
3. How to enforce reliability after the update?
4. How to protect intellectual property?

After defining a generic model of an update business process we discuss the security requirements imposed by the implementation of an upgrade option. We classify infrastructure requirements by upgrade data transport. Regarding the desire to integrate third party products and value-added functionality we finally discuss an open platform reliable system. In complex reliable systems the underlying technologies may also be vital in order to yield a testable system even if no upgrade functionality is foreseen.

2 Update Business Process

The business process for software updates must cover the software life-cycle. It should be designed such that it covers the following aspects:

1. Software development including change specification and final testing.

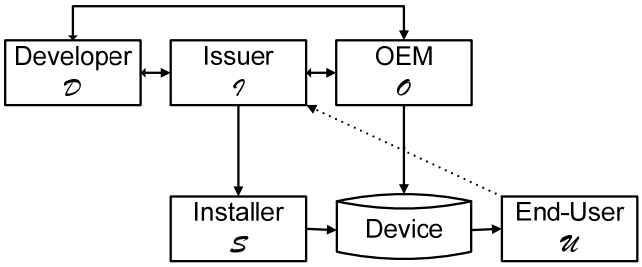


Figure 1: Roles in the business process. The end user buys the device from the OEM and contracts for upgrades with the issuer.

2. Maintenance of compulsory approvals for both functional aspects and security requirements, if applicable.
3. Distribution of the update to the customer and installation. This also includes licensing and payment aspects.
4. Deactivation or destruction of software with respect to expired licenses or security breaches.

The business process must coordinate the activities of several roles as sketched in figure 1. The relevant roles can be structured as listed below:

I: The software **issuer** manages the distribution of the updates to the installation sites and all licensing aspects. In case that the software requires deactivation or destruction, the issuer is responsible for enforcing such.

O: The original equipment manufacturer (**OEM**) creates the hardware and distributes it with initial software as supplied by the issuer.

D: The software **developer** creates the original software according to specifications supplied by the issuer and the OEM. The developer also produces update data according to specifications supplied by the issuer or the developer himself. The latter case mostly applies to bug fixes, the former to change requests and value added services.

S: The **installer** installs the update delivered by the issuer into the embedded system. The installer may be service personnel, the customer himself, an automatic procedure built into the system, or in most cases some combination of all of it.

U: The **end user**, who pays the issuer for the particular service or relies on the system functionality for his safety.

The list above has been sorted in order to avoid using undefined roles. This order reveals that the business process is determined by *I*, who should therefore own the process. Hence, *I* is responsible to run the largest part of the process' infrastructure. The infrastructure is significantly determined by the distribution method chosen by the issuer (sec. 4). The infrastructure in turn defines its necessary technical support in hard- and software, and should be specified before development begins; ideally even before *O* starts developing the hardware.

Although this key role of *I* for the upgrade business process appears evident, it is found as process owner only if software updates form the key business model, which is rarely the case with reliable systems. In security and safety markets this role often is even undefined or postponed for later definition. *I* should diligently examine, if the business case for the reliable system does actually exist. When postponing the definition of *I*, it should be considered that implementing the option for field updates produces considerable cost, even if the option is never used.

Parallels to the PC market are mostly deceiving. Software in the PC market virtually cannot be tested, since the system configuration is unknown. This is not acceptable with reliable systems and the classical hot-fix should not be of key relevance for a business case.

Development of a static product is much cheaper than a product, which is updatable in the field. These additional costs also add to the upgrade price. Furthermore, reliable systems often undergo formal approvals. The upgrade frequency cannot exceed the approval cycle. The costs for the infrastructure are thus distributed to only a few upgrades and each upgrade will additionally cause approval costs.

An interface for software updates may be misused by attackers. Misuse may cause system failure. Even if liability can be mitigated, damage to the brand may be severe. Therefore, this interface necessarily introduces security requirements as discussed in section 3. These requirements in itself may exceed the original system requirements.

3 Security Requirements

Security deals with the protection of assets by management of trust. A trusted instance is any instance able to compromise assets, unless it plays by the rules. Therefore, security is no property of the product delivered, but of the entire integrated system and processes. *I* trusts that the device produced by *O* runs a well defined software produced by *D*. In general also *O* trusts *I* to correctly specify the software and test benches.

Security introduces the role of an attacker *A*, who will intentionally attempt to compromise assets. *A* may be anything from competitors attempting to obtain intellectual property to end users trying to modify the system for their own needs, e.g. car tuners in the automotive market.

Software assets can be easily sketched by imagining the effects of the device to malfunction or being entirely out of service. With respect to modern networked system architectures, such faults may have more severe impact than obvious at first sight. As an example imagine a CAN node assuming a dominant ID and flooding the bus with garbage.

3.1 Issuer and Infrastructure

The required level of security cannot be covered by a blanket policy. Instead the issuer has to determine the required objectives from the following factors:

1. Compulsory requirements enforced by legal authorities.
2. Liability issues in case of product failure or misuse.
3. Protection of intellectual property.
4. Policy and brand issues.

The analysis of the security needs may yield formal requirements imposed by legal authorities as well as a financial risk figure summarizing the commercial threats. Unless there are no formal requirements and the risk figure is negligible, the issuer is advised to protect the distribution of updates (sec. 4).

To devise a business process for protected updates, it is necessary to define, who shall be trusted to which extent. The most trusted instance should be some organizational instance of the issuer himself. This security architecture shall be implemented using technical and organizational measures, which are visible by protocols. Protocols may range from signing-in for entering some room to technical measures in hard- and software. Since numerous publications on specific protocols exist, this paper does not focus on particular protocols.

The issuer being commercially responsible shall set up a business process, which adequately distributes cost into development and launching cost, maintenance cost for running the infrastructure, and additional cost per product for supporting the security architecture. The optimal security architecture will adapt this cost frame to the estimated product sales and life-time. Beyond these immediate costs the issuer shall estimate the financial risk of any of the trusted instances to be compromised.

In order to assess whether the business case is sound, these full costs may be distributed on the estimated updates on top of the immediate cost to program and release a particular update. If the intended frequency of updates is low, the immediate programming costs are easily outweighed.

Whichever process the issuer sets-up, he will end responsible for at least the root anchor of trust, usually implemented as a root secret. The issuer must keep in mind that the confidentiality and in most cases also the availability of this root secret condenses all the risks formerly assessed and is therefore worth whatever maximum risk figure has been estimated. Security of this key asset should be designed as if the corresponding amount in cash were to be stored. Equally, delegation of trust should be regarded as distribution of this amount. The infrastructure shall be reconsidered, if the cost saving by this delegation does not exceed the qualified risk figure.

Independent from the infrastructure of the delivery process $\mathcal{I}$ has to delegate the entire trust to $\mathcal{D}$. If the final product offers back-doors or weakness due to wrong implementation of protocols, this may cause the entire infrastructure to fail. Therefore, the trust delegated to $\mathcal{D}$ shall be valued as much as the anchor of trust controlled by $\mathcal{I}$ himself.

3.2 Software Development

The developer is trusted to neither intentionally nor negligently compromise the infrastructure ran by the issuer. In fact, there is no process to avoid this trust except product testing and review.

Testing and reviewing a product for security requirements is different from functional tests. As a rule of thumb, functional testing shall demonstrate that the product offers some known functionality. Security testing shall demonstrate that the product does not offer unknown functionalities.

The effort of security testing generally scales with the complexity of the product specification. This assumes that the product is designed to be testable with respect to security requirements, which in most cases requires a rigidly defined modular design with module boundaries cleanly reproducing the security protocols (sec. 5.1). It is not economically feasible to perform a black-box test on any modern IT product, in order to yield a qualified statement concerning product security.

In case the issuer and the developer are different entities or the issuer is not determined about the security implementation balancing the risk figure, it is strongly recommended to include a trusted third party, e.g. an accredited test laboratory, in order to mitigate the unlimited trust the issuer must delegate to the developer. Security certification schemes reflect the requirements for hierarchical trust management down to the implementation of protocols. A security certificate is one of the standard instruments to make design and implementation quality transparent to the issuer.

Using conventional system design, the rigid modular structure of the product limits the choices of updates beyond bug fixes. Actually, any type of imagined extension of the product must be considered during initial product design in order to have adequate structures available, when the extension is finally to be loaded into the device. This may be achieved by implementing well defined pluggable interfaces to connect future extension to. Such implementations reflect a security domain model as discussed in section 5.1.

Since $\mathcal{I}$ provides the specification to $\mathcal{D}$ it must define potential add-on upgrades before development. The choice of features to include in the business case is rarely extensible once the devices are in the field. Furthermore, as shown in section 5 software considerations may give rise to hardware requirements.

There is no test plan, which is suitable to reveal intentional loopholes in the software, particularly if the test plan is

known in advance. Software development of trusted components should be performed by trusted personnel in adequately secured environment. This simple statement clearly limits the choice of outsourcing decisions and thereby may raise the cost of a complex product considerably. Again, imagine the pile of cash from your risk estimation lying on some table at the development site. Furthermore, the costs for changing the developer throughout the product life-cycle are biased at least by the risk induced by broadening the delegated trust. Even after switching to a new developer the old developer may keep adequate knowledge of your product to significantly compromise trust.

3.3 Revision Control Infrastructure

As mentioned in section 1, update processes with reliable systems differ from updates in the PC market in that the system configuration may be completely known. This is a crucial advantage, if the system is expected to stay reliable after the update.

Actually, this insight is well known in the PC market yielding the two standard update models: online updates with the issuer querying the detailed system configuration before updating, and complete image updates entirely replacing previous versions.

Since online updates require a two way communication (sec. 4.2) the complete image method has been frequently selected (sec. 4.1). However, in practice this method assumes that data formats and memory layouts remain constant throughout the entire product life-cycle. Also, the required bandwidth, i.e. update time and resources, grows linearly with system complexity. Worse, if the configuration is not well defined, the software for the reliable system shall be tested with all possible configurations, no matter how senseless they may appear at first.

As soon as version or configuration specific parts are included in the update, e.g. a data conversion tool, the update shall be considered as partial, even if the entire program code of the embedded system is replaced as one single block. With respect to the required security infrastructure there is no difference to an update that attempts to patch some single byte in a well defined binary. The update procedure shall be robust with respect to update data, which were originally considered for a different product configuration or version. Even if all relevant updates have been distributed to all end users, it cannot be assumed that the updates were actually installed.

In order to support partial updates $\mathcal{I}$ and $\mathcal{D}$ must enforce a release process. Release testing shall include the update process, which in turn requires an appropriate archiving process for software and specifications.

With rising frequency of upgrades or configuration diversity of the target systems, the robustness of the platform with respect to inadequate updates must be increased. The corresponding mechanisms shall exist in the originally deployed version already.

4 Distribution and Infrastructure

A key issue with the distribution of partial upgrades and upgrades, which are intended for a limited selection of end users $\mathcal{U}$, is the method of distribution. Since distribution channels have a quite generic relation to the infrastructure to be implemented by $\mathcal{I}$, we discuss this aspect in more detail.

Distribution channels are classified into one-way communication $\mathcal{I} \rightarrow \mathcal{U}$, which we call *broadcasting* disregarding the number of immediately addressed instances of $\mathcal{U}$, and two-way communication. The latter falls into *online* methods, which establish a synchronous communication $\mathcal{I} \leftrightarrow \mathcal{U}$, and *offline* communication, which implement some kind of stored back-channel characterized by $\mathcal{I} \rightarrow \mathcal{S} \leftrightarrow \mathcal{U}$. We show that for critical updates in the field there are virtually no sensible alternatives to end-to-end security $\mathcal{I} \leftrightarrow \mathcal{U}$. In particular, all second best choices produce about the same overhead for infrastructure.

4.1 Broadcasting methods

In **simple broadcasting** distribution $\mathcal{I}$ sends the same product to all customers $\mathcal{U}$. It does not matter, if the information is actually sent to all $\mathcal{U}$ or a selection of $\mathcal{U}$. Provided that $\mathcal{U}$ are no instances of trust, the information is likely to be copied. From a security point of view, it virtually makes no difference, whether 10% of the customers receive a CD or all customers receive the update by satellite down-link. Chances are near certainty that at least one customer will allow someone to copy the CD. On the other hand, apart from a customer database in case of sending CD per mail, there is not much infrastructure required to implement the simple broadcasting distribution.

At the first glance, broadcasting appears as the distribution of choice for elementary bug fix updates. Considering reliable embedded systems and the corresponding business models there are obvious backsides.

Elementary bug fixes without the need of subscription are mostly distributed for functions, which have immediate impact on product liability. Relying on $\mathcal{U}$ to actively install an update available immediately puts $\mathcal{U}$ on the list of trusted instances and he will become part of the risk analysis of the issuer. This may be solved by enforcing automatic upgrades, which on the other hand requires some transmission of information without customer interaction, e.g. a GSM receiver. Since with broadcasting there is no back-channel, the requirements for tolerance concerning wrong versions are high. This may lead to systems refusing any further critical upgrade once they missed a revision for some reason.

In case the software contains commercial assets, attackers might use this broadcast information for compromising the latter. Reverse engineering of updates or modification of updates open up a variety of attack paths. Effective protection against modification of update data already requires simple cryptographic infrastructures enabling more sophisticated distribution methods.

Upon closer analysis the simple broadcasting method is only suited for value added services running in so called sandboxes (sec. 5.2), i.e. all kind of malfunction will not impact the function of the reliable host device.

The next more sophisticated method may be called **protected broadcasting**. This distribution method ensures that the update data can neither be reverse engineered nor be compromised in their integrity. This is often achieved by encrypting and signing the update data.

At least in order to decrypt the update data, the product must contain some kind of secret key. It does not matter whether this key is immediately used to decipher the data or to derive or import the actual decryption key. Simply from the fact that with broadcasting distribution all devices receive the same data, consequentially all devices must contain the same secret, i.e. a trusted system wide key. Attackers may be able to extract this key from any device deployed in the field, even if it must be severely damaged for the attack to succeed. Furthermore, such a key may leak from the developer or issuer site, unless properly protected.

Experience in security engineering shows that any system wide key cannot be properly protected in the field without bespoke hard- and software, if attackers estimate the market worth to perform sophisticated attacks. The issuer should take into account that the attackers' market is the same as his and offers fractions of the intended profits as funds for performing attacks. The simple infrastructure of protected broadcasting is always paid with unit cost for key protection. However, if reverse engineering of upgrade data may be tolerated, asymmetric cryptography schemes may enable bug-fix style upgrades on more standard hardware.

Putting this all together it seems sensible to use device keys for **individual broadcasting**. In individual broadcasting all devices receive different information. Depending on the communication channel there are two sub-categories of this scheme; the devices will either receive entirely different information or will receive a common block and only a minor individual part, e.g. an individually encrypted decryption key.

Common to both schemes $\mathcal{I}$ has to maintain a key infrastructure, which must, of course, be synchronized with the customer database. Effectively, the issuer must implement device tracking and must store or derive the individual secrets. This database in turn must be adequately protected throughout the entire product life-cycle. The infrastructure only puts the assets under control of $\mathcal{I}$ instead of $\mathcal{U}$ as all previous broadcasting schemes.

If only key information is sent individualized, the individual broadcasting reflects the common pay-TV architecture. The vulnerability is well known. If any of the receivers can be made to leak the plain text, the plain text is known for all devices in the field. For the embedded market this means that hacking a single device will compromise the update key for each update batch. On the other hand hacking the device in general must not destroy it, as was acceptable for retrieving the system wide keys used with protected broadcasting.

If the entire information is individualized, all update messages for each customer must be created and transported separately. Simple media like CD are rather unsuited for this type of distribution. Depending on the data size even online distribution may run into severe bandwidth limitations. This is why pay-TV implements partially individualized broadcasting.

Individualized broadcasting, if implemented correctly, offers a rock solid business process even for high-security applications. Attacks to devices in the field or the update data are virtually useless and attacks on system level can only originate from $\mathcal{I}$. However, $\mathcal{I}$ is required to run an infrastructure, which is comparable to two-way communication schemes.

4.2 Online methods (synchronous communication)

Online methods comprise requiring the customer calling some hotline in order to retrieve an activation code, up to online wireless protocols, which may be entirely hidden from the customer. From the security point of view the existence of some back communication channel enables to authenticate the end user and to retrieve configuration information, which is impossible in broadcasting schemes.

The infrastructure required to be run by the issuer does not differ significantly from individualized broadcasting. In order to authenticate end user, they must be registered and equipped with individual secrets.

The major advantage is constituted by the time of control of copies. When using broadcasting methods, attacks using illegal copies can at best be identified in case of suspicion by post-mortem forensics. This in turn requires a considerable robustness of the system's security. Online methods offer to control the product state at least at the point of updating, and if properly implemented require an attacker to communicate with the issuer's back-end system. This enables $\mathcal{I}$ to react much more flexible to new threats. The back-end system is, furthermore, able to gather any relevant auditing information in order to identify points of attack and to counteract by legal and organizational means.

Effectively online methods allow adapting the trusted instances throughout the life-cycle of the product by using trusted communication channels. The design of an infrastructure must enforce that this power for adaptation remains at the issuer. Most of the remaining potential compromise of trust can be reacted to on demand. This of course can significantly simplify product level protocols and therefore reduce product cost.

This saving of unit cost on the other hand generates cost for the infrastructure. The back-end system must allow end users to connect, which in turn may introduce vulnerabilities immediately to the trusted back-end system. Since the benefit arises from auditing and risk analysis in the back-end system, it must be implemented.

Obvious benefits exist, if $\mathcal{U}$ has many equally licensed products. This is particularly interesting, if the online communication is established with service personnel $\mathcal{I} \leftrightarrow \mathcal{S}$. In this case the infrastructure can be of small scale and $\mathcal{I}$

implicitly delegates trust to $\mathcal{S}$ to correctly use the update information. It shall be considered that the risk rises exponentially with the number of trusted $\mathcal{S}$ or $\mathcal{U}$, respectively.

Online methods are explicitly the method of choice, if the devices deployed in the field are expected to be equipped with a large number of different configurations, which have impact on the data required for updating. Such methods have been implemented and tested for a decent period of time in the PC market, e.g. for updates of operating systems (sec. 5.4). In this case the devices send the configuration data to the back-end system, which in turn compiles dedicated update data for the particular device. End-to-end security using effective authentication and encryption schemes is easily implemented and ensures that the back-end system is aware of the actual configuration of all devices deployed in the field.

4.3 Offline methods (asynchronous communication)

Offline methods are a generalization of online methods. With offline methods there is no immediate bidirectional communication channel $\mathcal{I}\leftrightarrow\mathcal{U}$, but $\mathcal{S}$ performs the update locally at the end user. This trusted instance uses some stored online information such that the trust is limited to the amount of online information stored.

$\mathcal{S}$ may range from service personnel supplied by $\mathcal{I}$ or a trusted workshop to some kind of smart media secure storage. The idea is to chain two or more online connections to finally build a trusted path $\mathcal{I}\leftrightarrow\mathcal{S}_i\leftrightarrow\mathcal{U}$. The trust delegated to $\mathcal{S}_i$ shall decrease with rising $i\leq f$. The final $\mathcal{S}_f$ is equipped only with the information necessary to perform the final online connection with $\mathcal{U}$, or decline the update, if the information is insufficient. In fact, completely individualized broadcasting (see section 4.1) may be considered as an offline method. However, we shall consider the mutual authentication as pre-requisite of offline methods, i.e. $\mathcal{S}_f$ must be able to withhold any information, if the authenticity of $\mathcal{U}$ cannot be established.

Offline methods are extremely powerful especially with a large number of end users in order to distribute the work load or bandwidth. Offline infrastructures can be designed to update millions of devices within a couple of hours. The infrastructure on the other hand comprises all secondary systems and their protocols and may even comprise the development of adequate smart media.

The discussion of offline infrastructures, which are tightly bound to specific business processes, is beyond the scope of this article. The general idea is similar to online schemes and the effective end-to-end security $\mathcal{I}\leftrightarrow\mathcal{U}$ provides a major design goal.

5 Updating complex systems

Reliable systems are often networked modules as components of some encompassing at least partially reliable system. A key feature of reliable systems is that these do not rely on external systems. This, too, is a security requirement in that reliable systems must not be perturbed in their functionality disregarding misuse of external interfaces. Security engineering mandates that no assumptions may be made concerning the data supplied on the interfaces and still the system shall be reliable.

In practice, assumptions are frequently implied. In this case, the external configuration of the encompassing system becomes part of the configuration to be considered for the upgrade. The design shall ensure that third party devices are not required to deliver critical information. Critical configuration information imported from other devices shall be protected or are put to the risk of alteration by $\mathcal{A}$.

The same concept applies for an entirely different task: the integration of non-critical value added applications onto a reliable platform. In particular, where complex reliable systems are deployed in mass markets such integration is often the dominant driver for the implementation of field upgrades.

In section 5.1 we define this concept of security domains, and discuss their application to upgrade processes in the following subsections of section 5.

5.1 Security Domains

A security domain D is a confined physical and logical unit where a single and homogeneous security policy is valid and applied. This security policy controls the security behaviour of application services A_i^D being provided in the context of this security domain.

The main generic characteristics of a security domain D with applications A_i^D and $i < n_D \in \mathbb{N}$ are the following:

- a security domain as a whole represents an encapsulated unit and can be considered as an *object*;
- internal and externally visible actions and reactions of this unit represent its well-defined *properties* (see also test aspects in section 3.2);
- communication between such objects occurs by well-defined (i.e. syntactic and semantic) messages and implements the *relationships* between the objects.

In case a security domain comprises more than one application, i.e. $n_D>1$ and direct communication is possible, e.g. $A_1^3\leftrightarrow A_2^3$ in figure 2, these applications shall be considered as a single application, whereas if this communication is impossible, e.g. $A_1^4\nleftrightarrow A_2^4$, these applications are separated in the sense used in this paper.

Since $A_i^D\nleftrightarrow A_j^E$ is a core requirement for separation of D and E, obviously the separating instance must be granted extended access rights with respect to the applications.

In case of physical separation all communication passes through the external bus and the requirement is trivial. This also holds, if the external bus is integrated into a multi-core micro-controller. Figure 2 A_1^1 depicts a physically separated application.

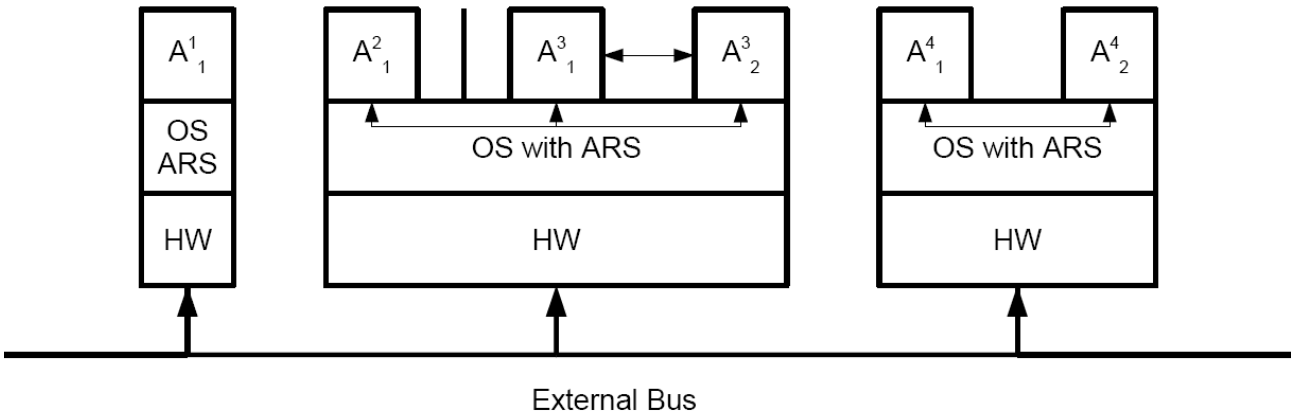


Figure 2: Implementation of security domains by physical separation A^1_1 and A^4_i , and logical separation A^2_1 and A^3_i in an environment of networked modules.

In case of logical separation this requires the OS to be of a different nature than A_i^D. This may be achieved by interpreting OS kernels, e.g. Java, or by hardware support, e.g. a supervisor mode and MMU support. Figure 2 A_1^2, A_i^3 depicts a logical separation scheme.

With respect to updates each security domain D running critical services A_i^D shall offer an application registration service (ARS). Update data shall include an application installation license (AIL), which is understood by the ARS. The ARS verifies the license, accepts or rejects the update data, and performs all necessary installation and configuration tasks depending on the AIL. The AIL shall contain the application identifier, version information (sec. 3.3), and the domain identifier (sec. 3.1), where the application has to be installed. The AIL may furthermore contain security certificates (sec. 3.2), (re-)configuration information for the security domain, copyright and usage restrictions, etc. Finally, an AIL shall be uniquely associated with the update data to process, e.g. contain a cryptographic hash, and all these data shall be digitally signed with respect to the issuer key associated with the security domain in order to provide end-to-end security (sec. 4).

If there is more than one application within same security domain ($n_D > 1$), and communication $A_i^D \leftrightarrow A_j^D$ is possible, the interaction between each pair A_i^D, A_j^D must be analyzed. Therefore, if an application shall often be updated, it is recommended placing it solely within a security domain ($n_D = 1$) in order to avoid this analysis. This model is considered in section 5.3.

While physical separation requires more hardware, the hardware and especially the operating system may be much less complex. The design of multi-application OS is in itself complex and may induce considerable requirements to the hardware (MMU, performance, etc.). Both, physical and logical separation are viable. However, if requirements shall consider the attacker role, there are hardly any out-of-the-box solutions currently available.

5.2 Application separation

With the growing power of embedded systems, designers have integrated various services A_i^D on the same hardware. These applications are rarely separated in the way depicted in figure 2 A_i^4, but often are even interwoven with the OS.

The potential interactions in between several tasks increase the system's complexity quadratically. In the end, it is simply impossible to predict, how the system will react to untested parameters, and by the same reason it is also impossible to test the system for all parameters. When considering upgrades, varying versions of A_i^D add another dimension to complexity. In order to retrieve any valid statement about failure risks, both functional and security risks, it is therefore mandatory to technically reduce the complexity of the system.

At this point functional requirements of $\mathcal{O}$ and $\mathcal{D}$, and security requirements of $\mathcal{I}$ converge. For $\mathcal{I}$ it is vital to test an update with respect to the final product configuration (sec. 3.3). If the separation is effective, each object can be tested by its properties. Upgrades leaving the properties unchanged or adding new objects will have no impact on the remaining system.

We stress that implementing applications as objects is a pre-requisite for versatile embedded systems, in particular if weakly controlled applications are considered. The design of an operating system supplying so-called sandboxes for objects is a complex task and is likely to be economically inadequate for any single device. Hence, the decision for the implementation of a logically sandboxed system shall be of strategic nature rather than be governed by operative desires. Physical separation may be more suitable, if the count of security domains is small.

5.3 Trust separation

With complex applications it is rarely the case that any part of the application can leverage the entire risk associated with the business process. In general, major parts of the application have virtually no impact on the assets at risk. The parts imposing security requirements are usually smaller and change at much slower frequency than value-added end user features. In vehicles the calculation of the

current speed is likely to never change, the maps used for the navigator will change regularly. The business case for frequent, individual updates is rarely found in critical functionality.

However, using the standard approach to embedded programming this insight does not immediately lead to benefits. Any change in non-critical functionality has the power to affect all critical functionality, e.g. an ill defined stop criterion while calculating routes in the navigator might inhibit speed information to the motion control system; left alone program crashes with uncontrolled memory access.

If the critical (trusted) parts run in a different security domain than non-critical parts, cost for releasing updates for the latter may drop to the immediately obvious cost of implementation and functional testing. This particularly applies to systems subject to compulsory requirements. A proper separation of security relevant parts from the rest may enable update processes, which do not require official reassessment.

5.4 Updating OS

So far the ARS was considered to install new applications A_i^D according to an AIL. What if the OS core with the ARS shall itself be updated? The corresponding vulnerability is again well known from the PC market as root-kits, which put the entire infrastructure at risk.

Severe damage of brand names and a considerable rate of licensing fraud has forced the big players in the PC market to consider alternatives. The major outcome are structures, which are summarized under the term *trusted computing*.

Trusted computing describes schemes for authentication of software layers rooting in a primary hardware layer. This layered structure is the key design feature of trusted computing platforms (TCP). In summary, trusted computing architectures put the system under control of $\mathcal{I}$ instead of $\mathcal{U}$, which is a main cause for the vivid discussion, since the PC is considered a multi-purpose device owned and consequentially controlled by $\mathcal{U}$.

Although the trusted computing approach to PC systems is questionable, it exactly fits the requirements for embedded applications in reliable devices. In general, $\mathcal{U}$ does not request full control concerning the configuration of e.g. his ABS, but he expects a vehicle to reliably stop. The idea of trusted computing should therefore be considered as a blueprint for designing more or less versatile, updatable embedded products.

A fundamental conclusion from trusted computing schemes is that the anchor of trust per device shall be located in the hardware. This is fairly easy to understand, because if the software is changed, the anchor of trust is lost as well. There is no system effectively able to verify its own integrity or existence, if any of the latter cannot be assumed in advance. On the other hand, if the hardware is changed, obviously an entirely different product has been substituted. Still, in a networked environment, this kind of substitution can be recognized and counteracted.

In case of integration of networked modules of the same security domain there shall be a root module, which will authenticate and enable further modules of the same primary security domain, e.g. constituted by $\mathcal{O}, \mathcal{I}$. This root module in turn shall have its anchor of trust implemented in hardware.

6 Conclusion

The integration of a field-upgrade option into a reliable system greatly exceeds the efforts for the immediate functional implementation. Overhead for implementation of upgrade support and the corresponding infrastructure shall be considered. This overhead often exceeds the immediate cost for the system without upgrade option.

If field-upgrades are to be supported, these shall be regarded as a business process with its own cost and risk management. The role of the *upgrade issuer* as a commercially responsible entity shall be defined, before the actual product development begins. The issuer shall devise a business process defining the trust management and in turn the technical and organizational requirements for its implementation. He shall maintain the anchor of trust during the entire product life-time. It is advised to diligently compare the total cost of an upgrade business process to conventional product replacement.

If critical functionality is involved, the infrastructure for upgrade distribution shall ensure end-to-end security. If reverse-engineering using upgrade data shall not be tolerated, this will require a two-way communication between the issuer and the end-user for each update, or bespoke hardware.

A domain model using trusted computing principles may be considered as a common prototype for all types of update policies. Implementation costs of such a platform are estimated to exceed the development budget of any single product and shall be rated as a strategic decision. There are similar models for layers of trust in systems constituted by networked modules.

Enforcing trust separation by a domain model may avoid compulsory (re-)assessments of software, if uncritical functionality is upgraded, only. Without trust separation the overhead for compulsory assessments shall be considered in the business process before deciding in favour of an upgrade option.

In any trust management scheme the *software and hardware developers* constitute entities trusted unconditionally. Diligent choice of developers and external review can greatly reduce the corresponding risks, which are beyond the control of the update issuer.

13th International Real-Time Ada Workshop

17-19 April 2007
Woodstock, Vermont
USA

Session: Language issues

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: Language Issues

Chair: T. Vardanega

Rapporteur: J. F. Ruiz

1 Introduction

This session focused on discussing open issues that arose from consideration of the real-time part of the Ada 2005 LRM. Three issues were table for the workshop to discuss. Accordingly, the goals of the session were to:

- Agree on proposed correction to EDF semantics
- Agree on resolution to requeue problem
- Review Ravenscar extensions for 2005 and consider extensions for distributed systems

2 Correcting the EDF definition in Ada 2005

The first part of this session addressed a problem in the current description of the Earliest Deadline First (EDF) dispatching protocol in the Ada 2005 standard. The difficult part in the definition of the EDF policy is the description of Ted Baker's Stack Resource Policy (SRP) for resource sharing [1].

The problematic wording is in the definition of the rules of EDF dispatching [3, D.2.6]. In that clause, the active priority of a task T when first activated or while it is blocked is defined as the maximum of the following:

- 24/2 the lowest priority in the range specified as *EDF_Across_Priorities* that includes the base priority of T,
- 25/2 the priorities, if any, currently inherited by T,
- 26/2 the highest priority P, if any, less than the base priority of T such that one or more tasks are executing within a protected object with ceiling priority P and task T has an earlier deadline than all such tasks.

Clause 26/2 contains the key semantics, and it is intended to ensure the behavior required by Baker's algorithm. Its current wording implies that a task could be placed on a ready queue above one on which a task with a shorter deadline is placed. Alan Burns described one scenario (the full details are in their position paper [2]) showing that the current wording is not in full accordance with Baker's protocol.

Alan Burns then illustrated a simple rewording to 26/2 which makes the rule stricter and achieves correct EDF dispatching in full adherence with Baker's SRP algorithm. The proposed rewording is as follows:

- 26/2* the highest priority P, if any, less than the base priority of T such that one or more tasks are executing within a protected object with ceiling priority P and task T has an earlier deadline than all such tasks and all other tasks on ready queues with priorities strictly less than P.

The workshop agreed that the current wording in 26/2 is not correct, approved the proposed reworking of it and tasked Alan Burns (as a member of the ARG) to produce an Ada Issue (AI) to illustrate the problem and propose the fix to it.

Michael González Harbour proposed to use a model checker to ensure that the definition complies with the intended semantics.

3 Requeuing via interfaces

The second part of the session discussed the possibility of allowing, in some form, requeuing via synchronized interfaces.

Timed and conditional entry calls, as well as asynchronous transfers of control (ATC), can use an interface as the target and the triggering event respectively, so it seems natural to allow requeue to an interface, both for completeness and consistency.

Andy Wellings explained that, as stated in their position paper [4], if requeue to an interface were allowed there are four cases whose semantics need to be defined:

- Requeue to an entry. This would follow normal requeue semantics.
- Requeue to a function (inside or outside a protected object). This would be an error condition that can be caught at compile time, because there are no circumstances whereby a function in an interface can be implemented by an entry. This is similar to the illegal case where a function call is used as the target of a timed/conditional entry call or as a triggering event in a select-then-abort statement.
- Requeue to a protected procedure. If a protected procedure is used as the target of a timed/conditional entry call or as a triggering event in a select-then-abort statement then the protected procedure is executed immediately, as if it were an entry with a "when True" barrier. Semantics for requeuing would be the same.
- Requeue to a "regular" procedure. If a procedure is used as the target of a timed/conditional entry call or as a triggering event in a select-then-abort statement then the procedure is executed immediately. It would appear natural to do likewise when requeuing, but that

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

would mean that the procedure would be executed at the ceiling priority of the original protected type (or priority of the original rendezvous), which is wrong.

The workshop agreed that requeue through synchronized interfaces is a useful and desirable primitive and should be supported in Ada so as to further the integration between object orientation and concurrency in Ada.

It was also agreed that the definition of requeue to an interface should be consistent with the use of interfaces in timed and conditional entry calls and asynchronous transfers of control. Consequently:

- Requeuing to an entry should follow normal requeue semantics.
- Requeuing to a function should be an error condition that can be caught at compile time.
- Two possible solutions exist to address requeues to both protected and regular procedures, which are discussed below:
    - A static scheme in which procedures within an interface can be identified as being "implemented as an entry" (by means of either a pragma, an "entry" identifier, or any other allowable mechanism), and requeues would only be accepted to procedures marked in that manner (which can be statically detected at compile time). If this approach were adopted, the semantics of timed/conditional entry calls and the select-then abort statement might need to be revisited to make the operations illegal on procedure calls.
    - A dynamic scheme in which these calls are detected at run time, and then the calling task/protected objects accept statement/entry code's body is "completed, finalized and left" (see [3, 9.5.4, par. 7]) before the procedure is called.

The workshop agreed that the preferred option would be the static scheme. If this option were not viable then the dynamic scheme could be considered, if the implementation (run-time) overhead was found to be reasonable.

Andy Wellings will ask Javier Miranda to develop a prototype implementation to evaluate the impact of the considered options.

4 Distributed systems with Ravenscar

The final part of the session discussed extensions to the Ravenscar profile to address high-integrity distributed systems. The use of the Distributed Systems Annex (DSA) for high-integrity systems is very appealing because it is very easy to use and static analysis can be performed among partitions.

DSA has not been widely used in high-integrity systems because it is not real-time and, in the Ada 95 version, it uses *Ada.Streams* (which is not recommended for high-integrity systems). However, there are DSA implementations compliant with the Ravenscar profile (PolyORB [6] can be configured that way), real-time DSA (such as RT-GLADE [7]), and the new Ada 2005 standard does not require *Ada.Streams*. Hence, there are good foundations to build on, but high-integrity systems would require an additional set of restrictions to guarantee the required degree of predictability, efficiency, and simplicity.

Santiago Urueña then presented the list of restrictions proposed in their position paper [5], categorized into a set of mandatory and optional restrictions. The set of mandatory restrictions would be made up by the following:

- No remote access types. This would allow the static creation of every required connection for each remote operation. Stephen Michell commented that this restriction will also avoid problems with the variable size of attribute *'External_Tag*. He also indicated that we should probably go further and forbid remote types (because they require *Ada.Streams*). Andy Wellings pointed out that it must be taken into account that restricting this will imply disallowing object-oriented programming in distributed systems altogether, which seems beyond what a Ravenscar-like profile should sanction.
- No concurrent remote calls. It would ensure that no remote operation can be called while processing a past invocation, thereby simplifying response time analysis analysis. Michael González Harbour pointed out that research works exist which target distributed systems showing how to take into account request queues in the analysis, so this restriction would not be strictly needed. He indicated also that the RPC receivers could be made more visible so that they can be dimensioned by the user and included in the analysis.
- Coordinated elaboration of partitions. The distributed application would not start until all its partitions have been elaborated, thereby improving determinism.

A further set of optional restrictions was also proposed that would simplify the implementation and facilitate response time analysis: no synchronous communication, no variable size messages, and no remote nested calls.

The workshop felt that there is a need to address high integrity real-time distributed systems, and Ada is very well placed for that (it would be an interesting topic for next IRTAW).The workshop encouraged people to work on this topic in order to be able to define the list of requirements for such systems and the model to support them.

5 Summary

The following summarizes the positions taken by the workshop during this session:

- An Ada Issue (AI) should be produced to fix the definition of the EDF protocol.
- Requeue through synchronized interfaces should be supported in Ada. A static and a dynamic scheme were

proposed (consistent with the use of interfaces in timed and conditional entry calls and asynchronous transfers of control) that need to be evaluated.

- There is a need to investigate models to support high-integrity real-time distributed systems.

References

[1] Baker, T. P. *Stack-based scheduling of realtime processes*. In Real-Time Systems, 3(1), March 1991.

[2] Zerzelidis, A., Burns, A., Wellings, A. J. *Correcting the EDF protocol in Ada 2005*. (this issue).

[3] Taft, S.T., Duff, R.A., Brukardt, R.L., Ploedereder, E., Leroy, P., eds.: *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).

[4] Wellings, A. J., Burns, A. *Integrating OOP and Tasking – The missing requeue*. (this issue)

[5] Urueña, S., Zamorano, J. *Building High-Integrity Distributed System with Ravenscar Restrictions*. (this issue)

[6] Vergnaud, T., Hugues, J., Pautet, L., Kordon, F. *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 106119, Palma de Mallorca, Spain, June 2004. Springer Verlag.

[7] López Campos J., Gutiérrez, J. J., González Harbour, M. *The chance for Ada to support distribution and real-time in embedded systems.* In Proceedings of the 9th International Conference on Reliable Software Techologies Ada-Europe 2004 (RST'04), volume LNCS 3063, pages 91-105, Palma de Mallorca, Spain, June 2004. Springer Verlag.

Correcting the EDF protocol in Ada 2005

A. Zerzelidis, A. Burns, A.J. Wellings

Real-Time Systems Research Group, Department of Computer Science, University of York, UK.

Abstract

Earliest Deadline First (EDF) dispatching has been introduced into the Ada 2005 definition. Included in this definition is support for Baker's protocol for preemption level control over access to protected objects. Unfortunately the current model fails to implement all the situations covered by Baker's approach. A counter example is provided that illustrates this deficiency with the language as currently defined. A minor change to the language definition is proposed that removes the flaw.

1 Introduction

One of the major enhancements contained within the Ada 2005 definition, in terms of the language's support for real-time applications, is the introduction of Earliest Deadline First (EDF) dispatching. Ada is the first mainstream engineering language to support this scheduling algorithm. EDF has been proven to be the most effective such algorithm available, in the sense that if a set of tasks is schedulable on a single processor by any dispatching policy then it will also be schedulable by EDF. Support for EDF requires two language features:

- representation of deadline for a task,
- representation of preemption level for a protected object.

The first is obviously required; the second is the EDF equivalent of priority ceilings and allows protected objects to be 'shared' by multiple tasks [1].

A deadline is usually defined to be *absolute* if it refers to a specific (future) point in time; for example there is an absolute deadline on the completion of this paper by 12th January 2007. A *relative* deadline is one that is anchored to the current time: "We have one month to finish this paper". Obviously, if the current time is known then a relative deadline can be translated into an absolute one (and vice versa). Repetitive (periodic) tasks often have a static relative deadline (for example 10ms after release) but will have a different absolute deadline every time they are released. The EDF scheme is more accurately expressed as *earliest absolute deadline first*.

The need to support preemption levels comes from the application of Bakers's protocol for controlling access to shared entities (see following discussion). Unfortunately, the language's support for this protocol is not complete. There are situations in which the behaviour of an Ada program will not follow the protocol and will therefore perform sub-optimally.

In this paper we illustrate this problem with the language as currently defined, and show how the problem can be rectified with a simple change to the wording of one paragraph in the Reference Manual. First however, Baker's protocol is defined and in Section 3 the EDF language features of Ada 2005 are described.

2 Baker's Preemption Level Protocol for Protected Objects

One of the major features that Ada 95 provided is support for monitors via the introduction of the protected object. Although the rendezvous is a powerful synchronization and communication primitive, it does not easily lead to tight scheduling analysis. Rather a more asynchronous scheme is desirable for real-time applications. The protected object provides this form of communication. With this construct, and the use of fixed priorities for tasks and a priority ceiling protocol for protected objects, it is possible to predict worst-case completion times for tasks.

With standard fixed priority scheduling, *priority* is actually used for two distinct purposes:

- to control dispatching, and
- to facilitate an efficient and safe way of sharing protected data.

The latter is often known as the priority ceiling protocol (in Ada it is called the *ceiling locking policy*). In Baker's stack-based protocol, two distinct notions are introduced for these policies [1]:

- earliest deadline first to control dispatching[1],
- preemption levels to control the sharing of protected data.

With preemption levels (which is a very similar notion to priority), each task is assigned a static preemption level, and each protected object (shared unit) is assigned a ceiling value that is the maximum of the preemption levels of the tasks that call it. At run-time, a newly released task, T1 say, can preempt the currently running task, T2, if and only if:

- the absolute deadline of T1 is earlier (i.e. sooner) than the absolute deadline of T2, and
- the preemption level of T1 is higher than the ceiling preemption level of every locked protected object.

[1] His paper actually proposes a more general model of which EDF dispatching is an example.

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

With this protocol it is possible to show that, on a single processor, mutual exclusion (over the protected object) is ensured by the protocol itself (in a similar way to that delivered by fixed priority scheduling and ceiling priorities)[2]. Baker also showed, for the classic problem of scheduling a fixed set of periodic or sporadic tasks, that if preemption levels are assigned according to each task's relative deadline then a task can suffer at most a single block from any task with a longer deadline. Again this result is identical to that obtained for fixed priority scheduling. **Note preemption levels must be assigned inversely to relative deadline (the smaller the relative deadline, the higher the preemption level).**

3 Supporting EDF Scheduling

It is an anomaly that Ada 95's support for real-time does not extend as far as having a direct representation for 'deadline', but Ada 2005 has rectified this by providing the following package:

```
with Ada.Real_Time; with Ada.Task_Identification;
package Ada.Dispatching.EDF is
   subtype Deadline is Ada.Real_Time.Time;
   Default_Deadline : constant Deadline :=
          Ada.Real_Time.Time_Last;
   procedure Set_Deadline(
          D : in Deadline;
          T: in Ada.Task_Identification.Task_ID :=
               Ada.Task_Identification.Current_Task);
   procedure Delay_Until_And_Set_Deadline(
          Delay_Until_Time : in Ada.Real_Time.Time;
          Deadline_Offset : in Ada.Real_Time.Time_Span);
   function Get_Deadline(
          T : Ada.Task_Identification.Task_ID :=
             Ada.Task_Identification.Current_Task)
          return Deadline;
end Ada.Dispatching.EDF;
```

The meaning of most of the operations of this package should be clear. A call of Delay_Until_And_Set_Deadline delays the calling task until time Delay_Until_Time. When the task becomes runnable again it will have deadline Delay_Until_Time + Deadline_Offset. The inclusion of this procedure reflects a common task structure for periodic activity. Note all tasks are given a deadline -if one has not been explicitly set then Time_Last (the far distant future) is given as the default.

A pragma is also provided to give an initial relative deadline to a task to control dispatching during activation.

To keep language changes to a minimum, Ada 2005 does not attempt to define a new locking policy but uses the existing ceiling locking rules without change. Priority, as currently defined, is used to represent preemption levels. EDF scheduling is defined by a new dispatching policy.

```
pragma Task_Dispatching_Policy (EDF_Across_Priorities);
```

[2] This property requires that there are no suspensions inside the protected object – as is the case with Ada's protected objects.

In Ada 95, dispatching is defined by a model that has a number of ready queues, one per priority level. Each queue, for the standard model, is ordered in a FIFO manner. Tasks have a base priority (assigned by pragma Priority and changed, if desired, by the use of Dynamic_Priority.Set_Priority). They may also have a higher active priority, if they inherit such a value during, for example, a rendezvous or execution within a protected object. Preemptive behavior is enforced by requiring a context switch from the current running task, if there is a higher priority non-empty ready queue. This is known in Ada as a *Dispatching Point*. At any dispatching point, the current running task is returned to its ready queue and another task (or indeed the same task if appropriate) is taken from its ready queue and executed. It should be noted that this is an abstract model of the required behavior; an implementation does not need to deliver the required semantics in this way. This also applies to the new model defined below.

3.1 The Rules of EDF Dispatching

The basic preemption rule given earlier for Baker's protocol, whilst defining the fundamental behavior, does not give a complete model. For example, it is necessary to define what happens to a newly released task that is not entitled to preempt. A complete model, within the context of Ada's ready queues, is defined by the following rules from the Ada 2005 reference manual (see paragraphs 17/2 to 28/2 of section D.2.6 of the manual).

When EDF_Across_Priorities is specified for priority range Low..High all ready queues in this range are ordered by deadline. The task at the head of a queue is the one with the earliest deadline.

A task dispatching point occurs for the currently running task T to which policy EDF_Across_Priorities applies:

- *when a change to the deadline of T occurs;*
- *there is a task on the ready queue for the active priority of T with a deadline earlier than the deadline of T; or*
- *there is a non-empty ready queue for that processor with a higher priority than the active priority of the running task.*

In these cases, the currently running task is said to be preempted and is returned to the ready queue for its active priority.

For a task T to which policy EDF_Across_Priorities applies, the base priority is not a source of priority inheritance; the active priority when first activated or while it is blocked is defined as the maximum of the following:

- *the lowest priority in the range specified as EDF_Across_Priorities that includes the base priority of T;*
- *the priorities, if any, currently inherited by T;*
- ***the highest priority P, if any, less than the base priority of T such that one or more tasks are executing within a protected object with ceiling***

priority P and task T has an earlier deadline than all such tasks.

When a task T is first activated or becomes unblocked, it is added to the ready queue corresponding to this active priority. Until it becomes blocked again, the active priority of T remains no less than this value; it will exceed this value only while it is inheriting a higher priority.

When the setting of the base priority of a ready task takes effect and the new priority is in a range specified as EDF_Across_Priorities, the task is added to the ready queue corresponding to its new active priority, as determined above.

In addition, there is a rule (paragraph 30/2 of section D.2.6) that no protected object can have a ceiling of value Low – when EDF_Across_Priorities is specified for priority range Low..High.

Rule 26/2 (the one in bold in the above quote) contains the key semantics, and is intended to ensure the behaviour required by Baker's Algorithm. Whilst in most cases this is true there is a circumstance in which tasks can execute in the wrong order. This is illustrated by Scenario 2 in the following example.

Example behaviour

Consider the following scenarios. Remember in all of these descriptions, the running task is always returned to its ready queue whenever a task arrives. A task (possibly the same task) is then chosen to become the running task following the rules defined above.

The system contains four tasks: T1, T2, T3 and T4 and three resources that are implemented as protected objects: R1, R2 and R3. Table 1 defines the parameters of these entities (in this table, *D* is relative deadline, *L* is preemption level, *U* is the resources used, *t* is the arrival time and *A* the absolute deadline.

Task	D	L	U	t	A
T1	100	1	R1,R3	0	100
T2	80	2	R2,R3	2	82
T3	60	3	R2	4	64
T4	58	4	R1	8	66

Table 1 A task set

We are concerned with just a single invocation of each task. The arrival times have been chosen so that the tasks arrive in order of lowest preemption level task first etc. We assume all computation times are sufficient to cause the executions to overlap.

The resources are all used by more than one task, but only one at a time and hence the ceiling values of the resources are straightforward to calculate. For R1, it is used by T1 and T4; hence the ceiling preemption level is 4. For R2, it is used by T2 and T3; hence the ceiling value is 3. Finally, for R3, it is used by T1 and T2; the ceiling equals 2 (see Table 2).

Protected Object	Ceiling Value
R1	4
R2	3
R3	2

Table 2 Ceiling Values

To implement this set of tasks and resources will require ready queues at level 0 (value of *Low* in this example) and values up to 4. Scenario 1 runs through a possible behaviour of the tasks set and illustrates the correct behaviour of the Ada 2005 protocol.

Scenario 1

At time 0, T1 arrives. All ready queues are empty and all resources are free so T1 is placed in queue 0. It becomes the running task. This is illustrated in the following where 'Level' is the priority level, 'executing' is the name of the task that is currently executing, and 'Ready Queue' show the other non-blocked tasks in the system.

Level	Executing	Ready Queue
0	T1	

At time 2, T2 arrives and is added to ready queue 0 in front of T1 as it has a shorter absolute deadline. Now T2 is chosen for execution.

0	T2	T1

Assume at time 3, T2 calls R3. Its active priority will rise to 2.

2	T2	
0		T1

At time 4, T3 arrives. Task T2 is joined by T3 on queue 2, as T3 has an earlier deadline and a higher preemption level; T3 is at the head of this queue and becomes the running task.

2	T3	T2
0		T1

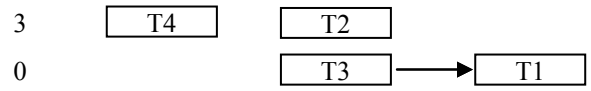
At time 8, T4 arrives. Tasks T3 and T2 are now joined by T4 as it has a deadline earlier than T2 and a higher preemption level (than 2). Task T3 remains the running task, and will execute until it completes.

2	T3	T4 →	T2
0		T1	

Scenario 2 -Incorrect Behaviour

Here we make the simple change that, at time 3, T2 calls R2 instead of R3. Its active priority will rise to 3. Now when T3 arrives at time 4, it will not have a high enough preemption level to join ready queue 3 and will be placed on the lowest queue at level 0 (but ahead of T1). Task T2 continues to execute.

At time 8, T4 arrives. It passes both elements of the test and is placed on the queue at level 3 ahead of T2 and therefore preempts it.



The result of this last modification is that T4 is executing even though T3 is in the system and has an earlier deadline than T4. This is not in accordance with Baker's protocol. To force compliance with the protocol the following change has to be made.

4 Correcting the Definition

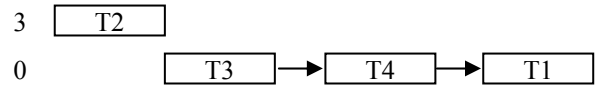
The problem with the current language definition is that Rule 26/2 is not strong enough: task T must have an earlier deadline than all tasks not only at level P but at all lower levels. So the final part of the rule to determine the active priority of a task (i.e. 26/2):

- *26/2 the highest priority P, if any, less than the base priority of T such that one or more tasks are executing within a protected object with ceiling priority P and task T has an earlier deadline than all such tasks.*

must become:

- *26/2* the highest priority P, if any, less than the base priority of T such that one or more tasks are executing within a protected object with ceiling priority P and task T has an earlier deadline than all such tasks and all other tasks on ready queues with priorities strictly less than P.*

Now Scenario 2 described above will behave as follows: at time 8, T4 will 'fail' 26/2* and will be placed on the level 0 queue between T3 and T1.



T2 will continue until it completes its execution in the protected object. Its priority will then fall to 0 and T3 will preempt it. All tasks now execute in deadline order – earliest first.

A formal proof of the revised protocol is not possible within the space allocated to this paper, however a key property can be explored via the following considerations. Strict EDF dispatching is ensured as long as a task T1 with an absolute deadline d1 is never on a ready queue below that of task T2 with absolute deadline d2 when d1<d2 unless T2 is using a shared resource and as a result has an inherited preemption level. This follows from the fact that ready queues are dispatched in priority order. There are two cases to consider:

Task T1 arrived before T2: T2 cannot be placed above T1 (in the ready queue order) if it arrived later and d1 is before d2 – direct consequence of 26/2*.

Task T1 arrived after T2: note, T2 has no inherited preemption level. As T1 has an earlier deadline than T2 it would be placed on the same queue as T2 (or higher) unless it has a lower preemption level. If it have a lower preemption level then we have the properties: T1 arrives after T2, has an earlier deadline than T2, but a lower preemption level. This is in direct contradiction to the preemption level allocation algorithm required by Baker (see discussion at the end of Section 2). Hence T1 cannot have a lower preemption level and would thus be placed on an equal or higher ready queue to that of T2.

5 Conclusion

The inclusion of EDF dispatching into the Ada language is a major enhancement. Ada becomes the first main stream engineering language to support this common protocol for controlling the execution of real-time activities. Operating systems, as well as languages, at best support only fixed priority dispatching. But for many resource restricted applications, the extra performance one can get from EDF scheduling is a major attraction. Ada's move to support EDF and fixed priority dispatching, and their integrated use within the same program will therefore open up the use of Ada in a number of new fields including mobile computing.

Although the definition contained within the Ada 2005 definition attempted to implement Baker's preemption-level protocol within the existing rules for priority ceiling locking for protected objects, the current model is not entirely satisfactory. The problem has been identified in this paper. Fortunately a minor rewording of one paragraph is all that is needed to remove the flaw and complete the protocol's definition.

Acknowledgements

The authors would like to thank Ted Baker for useful discussions on his protocol. Also John Barnes for comments on an earlier draft of the paper. The work reported in this paper is funded, in part, by the EU project ARTIST.

References

[1] T.P. Baker. *Stack-based scheduling of realtime processes*. Real-Time Systems, 3(1), March 1991.

Integrating OOP and Tasking – The missing requeue*

A. J. Wellings, A. Burns

Real-Time Systems Research Group, Department of Computer Science, University of York, UK; email: {andy,burns}@cs.york.ac.uk

Abstract

Ada 2005 has provided limited integration between its object-oriented programming and tasking facilities. The compromise has been to allow tasks and protected types to support interfaces but not inheritance. Whilst the language supports timed/conditional entry calls and the select-then-abort statement using interfaces, it does not allow the target of a requeue statement to be determined via an interface. This paper argues that this design decision needlessly limits the expressive power of the language. A motivating example is used to illustrate the case, and the semantics of the proposed language extension are discussed.

1 Introduction

One of the topics that has been covered by past workshops has been the integration of Ada 95 object-oriented programming features with the Ada tasking model [1, 2, 5]. Full integration would add significant complexity to the language (for example, see the suggested models [4, 8, 7, 6, 3]), and proposals for language changes did not receive widespread support. Ada 2005 has added Java-like interfaces to the OOP model and has used this opportunity to provide limited integration. The compromise is to allow task and protected types to implement some types of interfaces but still not to be tagged types. Hence inheritance is not supported but run-time dispatching is.

A side effect of the Ada 2005 approach is that it is now possible to make timed (or conditional) entry calls or use the select-then-abort statement whose target entries are based on normal Ada procedures (if they implement part of a limited, synchronized, protected or task interface). The language defines that when such a call is issued it is deemed to happen immediately and hence no timeout or wait occurs, and in the case of the select-then-abort statement, no abortable code is started. Whilst these side effects might seem strange, it is a consequence of the decision taken that interfaces that have some synchronization implied should be identified explicitly. In this way, the programmer knows that calls are potentially suspending and can add timeouts if necessary. It is also necessary to more easily identify potentially suspending calls from within a protected action. In contrast, Java adopts the approach where interfaces have no synchronization constraints identified, even though the encapsulated methods can be implemented by synchronized methods. But then Java does not provide an avoidance model for condition synchronization; instead it uses conditional waits within a synchronized method (where timeouts can be added if needed).

Although Ada 2005 has gone a long way to integrating its access to entries via interfaces, it has shied away from allowing a requeue operation via an interface. The following is taken from the minutes of the ARG Paris meeting (Feb. 2005) under a discussion on AI-397:

> "Steve Baird asks if you can requeue on a synchronized procedure. No, we don 't allow that, because the synchronized procedure doesn't necessarily denote an entry. And what such a requeue would do if the procedure wasn't an entry is rather unclear."

This paper suggests that this decision should be revisited. In section 2, we briefly review the Ada 2005 OOP and tasking integration, and examine the general role of requeue. In section 3, we present a motivating example for why requeue should be allowed to occur via a synchronized (or task or protected) interface. Then, in section 4, we define the semantics of such an operation. Finally, we draw our conclusions.

2 Tasking, OOP and Requeue

An interface is a type that has no state. It is essentially an abstract tagged typed with null components. All primitive operations on an interface type must be abstract or null.

Although the inspiration for Ada interfaces comes from the Java language, they cannot be used in exactly the same way as Java interfaces but should be thought of as abstract types. In particular, it is not possible to define an arbitrary subprogram that takes as a parameter an interface (as can be done in Java). The equivalent in Ada 2005 is to define the parameter type as a *class-wide type* or *class-wide access type* rooted at the Ada interface. Hence, Ada requires you to distinguish between a classwide type and a specific type. This has the advantage of allowing differentiation between dispatching calls and non-dispatching calls at the source level.

* This work has been undertaken within the context of the EU ARTIST2 project.

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

Formally, a tagged type can be derived from zero or one parent tagged type plus zero of more interface types. If a non-interface tagged type is present it must be the first in the list. Hence interfaces themselves can be derived from other interfaces.

2.1 Limited interfaces

In Ada, a limited type is a type that does not support the assignment statement and has no predefined equality operator. Interfaces can be limited; they are, however, not limited by default. A non-limited interface has a predefined equality operator available, whereas a limited interface does not.

A new interface can be composed of a mixture of limited and non limited interfaces but if any one of them is non limited then the resulting interface must be specified as **non limited**. This is because it must allow the equality and assignment operations implied by the non-limited interface.

Similar rules apply to derived types that implement one or more interfaces. The resulting type must be non limited if any of its associated interfaces are non limited.

2.2 Synchronized, protected and task interfaces

In order to provide integration between Ada's concurrency and OOP models, Ada 2005 provides three further forms of limited interfaces: *synchronized*, *protected* and *task*.

The key idea of a synchronized interface is that there is some implied synchronization between the task that calls an operation from an interface and the object that implements the interface.

Synchronization in Ada is achieved via two main mechanisms: the rendezvous and the protected action (call of an entry or protected subprogram). Hence, a task type or a protected type can implement a synchronized interface. The rendezvous provides control-oriented synchronization and the protected type provides data-oriented synchronization. Where the programmer is not concerned with the type of synchronization, a synchronized interface is the appropriate abstraction. For situations where the programmer requires control-oriented (or data-oriented) synchronization, task interfaces (or protected interfaces) should be used explicitly.

In summary:

- All task and protected interfaces are also synchronized interfaces, and all synchronized interfaces are also limited interfaces.
- A synchronized interface can be implemented by either a task or protected type.
- A protected interface can **only** be implemented by a protected type.
- A task interface can **only** be implemented by a task type.
- The controlling (dispatching) parameter in a synchronized (or protected or task) interface must be the first parameter. Furthermore, if the operation is to be implemented as an entry (either protected or task) then it should be an "out", an "in out" parameter or an "access" parameter.

There is a hierarchy in the types of limited interfaces: limited comes before synchronized, which comes before task/protected. Task and protected interfaces have equal position in the hierarchy. Two or more interfaces can be composed as long as the resulting interface is not before any of the component interfaces in the hierarchy. Hence a limited interface *cannot* be composed from a synchronized, protected or task interface. A synchronized interface cannot be composed from a task or protected interface; and a task or protected interface *cannot* be composed from each other.

Calling operations on objects that implement limited interfaces

Operations on interfaces that are defined as limited, synchronized, protected or task may be implemented directly by a protected or task type. The following should be noted.

1. A call to any operation on an object that implements a synchronized (or protected or task) interface may be blocked until the right conditions are in place for that operation to be executed. If the operation is implemented by a protected procedure or function then this blocking will be bounded. If the operation is implemented by a task or a protected entry then the blocking may be indefinite. Consequently, Ada 2005 allows a call to an operation defined by a synchronized (or protected or task) interface to be placed in a "timed or conditional entry call" statement or in a "select then abort" statement.
2. A task or protected type can also implement a limited interface. Consequently, call to objects that implement limited interfaces may also block! Ada 2005, therefore, allows them to be placed in a timed or conditional entry call statement or in a select-then-abort (ATC) statement.
3. Any call to an operation on an object that implements a limited (or synchronized or protected or task) interface that dispatches to a non-entry call, is deemed to have been "accepted" immediately and, therefore, can never time-out.

2.3 Tasks and interfaces

A task type in Ada 2005 can be declared to "implement" zero, one or more combinations of limited, synchronized and task interfaces. Task interfaces allow programmers to specify interfaces that must be implemented by tasks. Hence they allow the programmers greater control over expressing their intentions. However, as already mentioned, it is not possible for tasks to be derived from other tasks. Obtaining a semantic model for this is challenging given an already-existing language.

2.4 Protected types and interfaces

Protected interfaces are interfaces that can only be implemented by protected types. Protected interfaces should be used when

- the programmer wants data-oriented rather than control oriented synchronization,
- there is a level of indirection between the tasks needing the synchronization, or
- the programmer wishes to ensure that the required synchronization is implemented by a passive synchronisation agent rather than an active one.

2.5 Synchronized interfaces

Task and protected interfaces are the correct abstractions to use when the programmer wishes to commit to a particular implementation strategy. In some circumstances, this may not be appropriate. For example, there are various communication paradigms that all have at their heart some form of buffer. They, therefore, all have buffer-like operations in common. Some programs will use these paradigms and will not care whether the implementation uses a mailbox, a link or whatever. Some will require a task in the implementations, others will just need a protected object. Synchronized interfaces allow the programmer to defer the commitment to a particular paradigm and its implementation approach.

2.6 Requeue

Ada allows requeues between task entries and protected object entries. A requeue can be to the same entry, to another entry in the same unit, or to another unit altogether. Requeues from task entries to protected object entries (and vice versa) are allowed. However, the main use of requeue is to send the calling task to a different entry of the same unit from which the requeue was executed. It is important to appreciate that requeue is not a simple call. If procedure P calls procedure Q, then, after Q has finished, control is passed back to P. But if entry X requeues on entry Y, then control is not passed back to X. After Y has completed, control passes back to the object that called X. Hence, when an entry or accept body executes a requeue, that body is "completed, finalised and left" (ARM, Section 9.5.4).

One consequence of this is that when a requeue is from one protected object to another then mutual exclusion on the original object is given up once the task is queued. Other tasks waiting to enter the first object will be able to do so. However, a requeue to the same protected object will retain the mutual exclusion lock (if the target entry is open).

Finally, when requeuing from one protected object to another it is important to understand that while the call is being evaluated a mutual exclusion lock is held on both protected objects. It is therefore a bounded error to execute an external requeue request back to the requesting object (it would inevitably lead to deadlock). Moreover, the programmer cannot assume, when a requeue has taken place from one protected object to another, that any tasks released in the original object will execute before the entry in the destination object. Consider the following contrived example that consists of two protected objects.

```
protected Original is
  entry One;
  entry Two;
private
  Go : Boolean := False;
end Original;

protected Target is
  entry One;
end Target;
```

In the body of this, a call to Original.One is requeued to Target.One:

```
protected body Original is
  entry One when True is
  begin
    Go := True;
    requeue Target.One;
  end One;
  entry Two when Go is
  begin
    Go := False;
    Put_Line("Original Two executed");
  end Two;
end Original;

protected body Target is
  entry one when True is
  begin
    Put_Line("Target One executed");
  end One;
end Target;
```

Now consider a case where a task (A) is already queued on the Original.Two entry when another tasks (B) calls Original.One. As the barrier on Original.One is open, the entry is executed and Go flag is set to true, and the call is requeued on Target.One. First the lock on the Target protected object is obtained, and then the barrier is tested. If the barrier on Target.One was false, task B needs to be queued and then the Original.One entry needs to be finalized which involves task B executing the entry Original.Two on behalf on task A. Hence, the lock on the Target protected object is maintained until this finalization code has been executed; task B is then placed on the entry queue. If the barrier on Target.One is True (as it is in this case), task B executes the Target.One entry code and then finalizes that entry. It then finalizes the Original.One entry and executes the Target.One entry code and then finalizes that entry. Hence in the above example, the output "Target One executed" appears before "Original Two executed".

3 A Motivating Example

Although requeuing to the same entity represents the normal use of requeue, there are situations in which the full generality of this language feature is useful.

Consider the situation in which resources are controlled by a hierarchy of objects. For example, a network router might have a choice of three communication lines on which to forward messages: Line A is the preferred route, but if it becomes overloaded Line B can be used; if this also becomes overloaded Line C can be used. Each line is controlled by a server task; it is an active entity as it has housekeeping operations to perform. A protected unit acts as an interface to the router; it decides which of the three channels should be used and then uses requeue to pass the request to the appropriate server. The structure of the solution is given in the program fragment is given below:

```
type Line_Id is (Line_A, Line_B, Line_C);

type Line_Status is array (Line_Id) of Boolean;

task type Line_Controller(Id : Line_Id) is
  entry Request(...);
end Line_Controller;

protected Router is
  entry Send(...);
  procedure Overloaded(Line : Line_Id);
  procedure Clear(Line : Line_Id);
private
  Ok : Line_Status := (others => True);
end Router;

La : Line_Controller(Line_A);
Lb : Line_Controller(Line_B);
Lc : Line_Controller(Line_C);

task body Line_Controller is
  ...
begin
  loop
    select
      accept Request(...) do
        - - service request
      end Request;
      or
        terminate;
    end select;
    - -housekeeping including possibly
    Router.Overloaded(Id);
    - -or
    Router.Clear(Id);
  end loop;
end Line_Controller;

protected body Router is
  entry Send(...) when Ok(Line_A) or
                       Ok(Line_B) or Ok(Line_C) is
  begin
    if Ok(Line_A) then
      requeue La.Request with abort;
    elsif Ok(Line_B) then
      requeue Lb.Request with abort;
    else
      requeue Lc.Request with abort;
    end if;
  end Send;

  procedure Overloaded(Line : Line_Id) is
  begin
    Ok(Line) := False;
  end Overloaded;

  procedure Clear(Line : Line_Id) is
  begin
    Ok(Line) := True;
  end Clear;
end Router;
```

Now consider the generalization of code to allow a router to support *any* type of network interface controller. First, it is necessary to define the interface that all line controllers must support:

```
package Network_Interface is
  type Line_Interface is task interface;

  procedure Request(LI: in out Line_Interface)
        is abstract;

  type Any_Line_Interface is access all
        Line_Interface'Class;
end Network_Interface;
```

Any line controller can be defined, as long as it supports this interface:

```
type Line_Id is (Line_A, Line_B, Line_C);

type Line_Status is array (Line_Id) of Boolean;

task type Line_Controller(Id : Line_Id) is new
       Line_Interface with
    overriding entry Request(...);
end Line_Controller;
```

Now a general purpose router can be defined that will work with any line controller

```
protected type Router(LA: Any_Line_Interface;
                      LB: Any_Line_Interface;
                      LC: Any_Line_Interface) is
  entry Send(...);
  procedure Overloaded(Line : Line_Id);
  procedure Clear(Line : Line_Id);
private
  Ok : Line_Status := (others => True);
end Router;
```

However, the body of this protected type causes a problem:

```
protected body Router is
  entry Send(...) when Ok(Line_A) or
                       Ok(Line_B) or Ok(Line_C) is
  begin
    if Ok(Line_A) then
      requeue LA.Request with abort; - -** NOT LEGAL
    elsif Ok (Line_B) then
```

```
      requeue LB.Request with abort; - -** NOT LEGAL
    else
      requeue LC.Request with abort; - -** NOT LEGAL
    end if;
  end Send;
  - -as before
end Router;
```

What this example shows is two things:

1. Requeing from one task/protected type to another is an integral part of the Ada model and one that gives it great power and flexibility.
2. Requeuing via interfaces is a necessary facility if interfaces are to be fully integrated with tasks and protected types.

4 Proposed Revised Requeue Semantics

Section 3 has illustrated the need to allow the use of requeue with interfaces. In this section we consider the semantics for the situation where the target requeue is not an entry call. There are three possible situations.

1. The target is a function inside or outside a protected object – this is an error condition that can be caught at compile time; there are no circumstances whereby a function in an interface can be implemented by an entry. This is similar to the illegal case where a function call is used as the target of a timed/conditional entry call or as a triggering event in a select-then-abort statement.
2. The target is a procedure inside a protected object – this is similar to the case where the target of a timed/condition entry call or select-then-abort triggering event turns out to be a procedure that has been implemented inside of a protected type implementing the associated interface. The corresponding interpretation would be to view the procedure as an entry with a "when True" barrier. Hence, a new protected action is started for the target protected object and the procedure is executed immediately (ARM, Section 9.5.4 par 11).
3. The target is a regular procedure (i.e. outside a protected object/task) – this is similar to the case where the target of a timed/condition entry call or select-then-abort triggering event turns out to be a procedure outside of the protected object/tasks implementing the associated interface. As this would succeed in this case, so should it succeed in the requeue case. However, to execute the procedure immediately would mean that the procedure would be executed at the ceiling priority of the original protected type (or the priority of the original rendezvous). This would seem to be wrong. Hence, the current protected action continues and completes before the procedure is called.

Return again to the example given in Section 2.6, but this time suppose that the Target is defined as:

```
type Target is synchronized interface;
procedure One(T: in out Target) is abstract;
type Any_Target is access all Target'Class;
```

and the Original protected object is now a type parameterized by the target.

```
protected Original(Target : Any_Target) is
  entry one;
  entry two;
private
  Go : Boolean := False;
end Original;

protected Target is
  entry one;
end Target;

protected body Original is
  entry One when True is
  begin
    Go := True;
    requeue Target.one;
  end One;

  entry Two when Go is
  begin
    Go := False;
    Put_Line("Original Two executed");
  end Two;
end Original;
```

Assuming that the target subprogram again outputs the string "Target One executed", then if the actually target One turns out to be an open entry or a procedure then once again the output would be "Target One executed" followed by "Original Two executed". However, if the actual target One turns out to be a procedure outside of a protected type, the output would be "Original Two executed" followed by "Target One executed". These two cases would need to be recognised by the run-time system. It is unclear whether this would impose a significant implementation burden.

Alternative approach

As an alternative to the proposal given above, it could be argued that attempting to requeue to a non-entry is an error condition and a run-time exception (Program_Error) should be raised. As the Ada 2005 designers chose not to do this for the timed/condition entry call and select-then-abort cases, we proposed, for consistency, the same model.

Of course, where the programmer knows that the intended target is an entry, ideally there should be a mechanism to indicate this (perhaps by a pragma). This would allow the compiler to check, thereby avoiding the run-time error. If this approach were adopted, the semantics of timed/conditional entry calls and the select-then-abort statement might need to be revisited to make the operations illegal on procedure calls again. However, given that limited interfaces can also be implemented by tasks and protected objects, adding such a pragma would, in effect, make them synchronized.

5 Conclusions

This paper has argued that it should be possible to execute a requeue statement (both with and without abort) via a limited, synchronized, protected or task interface. The proposed semantics are consistent with those that are currently defined for conditional and timed entry calls, and for the use of interfaces with the select then abort statement. Indeed, given the current semantics there are obvious counterparts for the requeue case. However, an implementation must perform the finalization of a called protected entry in a different order if the target is not encapsulated within a protected object. The actual overhead that this incurs are implementation-dependent.

6 Acknowledgements

The authors wish to thank Tucker Taft for a comment on an early draft of this paper and for helping us refine the proposed semantics given in Section 4.

References

[1] J. de la Puente. *Session summary: Object-oriented programming and real-time.* In Proceedings of IRTAW8, Ada Letters, pages 11–15, 1997.

[2] J. A. de la Puente. *New language features and other language issues.* In Proceedings of IRTAW9, Ada Letters, Vol XIX(2), pages 19–20, 1999.

[3] R. Garcia and A. Strohmeier. *Experience report on the implementation of EPTs for GNAT*. In Proceedings of IRTAW11, Ada Letters, Vol XX1I(4),pages 22–17, 2002.

[4] O. P. Kiddle and A. J. Wellings. *Extended protected types*. In Proceedings of ACM SIGAda Annual International Conference (SIGAda 98), pages 229–239, November 1998.

[5] J. L. Tokar. *Tasking and object orientation*. In Proceedings of IRTAW10, Ada Letters, Vol XX1(1),pages 9–10, 2001.

[6] A. J. Wellings, B. Johnson, B. Sanden, J. Kienzle, T. Wolf, and S. Michell. *Integrating object-oriented programming and protected types in Ada 95*. ACM TOPLAS, 22(3):506–539, 2000.

[7] A. J. Wellings, B. Johnson, B. Sanden, J. Kienzle, T. Wolf, and S. Michell. *Object-oriented programming and protected objects in Ada 95.* Reliable Software Technologies - Ada-Europe 2000, Lecture Notes in Computer Science, 1845:16–28, 2000.

[8] A. J. Wellings, B. Johnson, B. Sanden, J. Kienzle, T. Wolf, and S. Michell. *Extensible protected types: Proposal status*. In Proceedings of IRTAW10, Ada Letters, Vol XX1(1), pages 105–110, 2001.

Building High-Integrity Distributed Systems with Ravenscar Restrictions*

Santiago Urueña, Juan Zamorano

Department of Computer Architecture and Technology (DATSI), Technical University of Madrid (UPM), E28660 Boadilla del Monte, Spain; e-mail: Santiago.Uruena@upm.es, JuanRafael.Zamorano@upm.es

Abstract

The Ravenscar profile was a qualitative leap in the development of single-processor hard real-time systems with certification requirements. But nowadays more and more safety-critical systems are distributed, so a new Adaprofile is needed for multi-node applications. This work discusses the restrictions and additions to the language needed to certify and obtain the required predictability and timeliness in a high-integrity hard real-time Ada distributed application.

1 Introduction

Nowadays, several types of High-Integrity Systems (HIS)—specially safety-critical ones— have hard real-time requirements and must execute in an embedded distributed hardware. A distributed system is not only needed when different parts of the system are physically distant. It also provides better fault-tolerance, isolates applications with different criticalities among the nodes, and gives more processing power than a single CPU.

The Ravenscar profile was designed for HIS with strict temporal requirements [6], and where response-time analysis (RTA) methods [10] are needed, easing the development and certification of those types of systems. It was one of the major additions to Ada 2005 [1], and there are multiple commercial implementations [3]. However, the profile was targeted for mono-processors and thus no special support was designed for the development of distributed systems.

Although the Distributed Systems Annex (DSA) is not forbidden in the Ravenscar profile defined in Ada 2005 [9, §D.13.1] it presents some problems that make this Annex E inadequate for hard real-time distributed systems [14]. This is not surprising because the DSA was designed for general purpose distributed systems, although the implementators are allowed to extend and adapt the annex with new functionality and rules [9, § E.2(14), E.2.1(8.b), E.5(26), E.5(27.1/2)].

Therefore, Ravenscar applications can currently use the DSA in a "portable" way only for non real-time communication with other nodes. The goal of this paper is to restrict the DSA following the philosophy of the Ravenscar profile to enable the certification of the middleware, to achieve the required degree of predictability and timeliness, and to discuss the most important additions needed for the development of distributed hard real-time safety-critical systems.

This paper covers distributed applications running on a physically distributed system (different computing nodes connected through a bus or a local network), or on the same node (logical partitions, like ARINC-653). The existence of a reliable hard real-time communication network is of paramount importance. However, this paper is not aimed at multi-processor systems, except if each partition can execute only in the same CPU (i.e. no 'task migration').

This paper is organized as follows. Section 2 describes the related work, including the scheduling theory. Section 3 discusses the advantages and disadvantages of different distribution mechanisms used by the software industry. Section 4 defines a set of restrictions for building safety-critical distributed systems, while section 5 talks about the implementation requirements. Section 6 presents two use cases. Finally, section 7 summarizes the main conclusions of this work.

2 Related Work

This paper builds upon current advances in scheduling theory for distributed hard real-time systems. Tindell and Clark [24] extended the responsetime analysis techniques used for single processors to distributed systems, introducing the concept of holistic schedulability. Later, Palencia and González Harbour [16] improved the technique to reduce the pessimism of transactions.

Work about adapting the Ada distributed system annex for real-time systems includes past IRTAW sessions [14][7] and two different implementations: RT-GLADE [13], a research real-timeimplementation of the DSA, and PolyORB [25], a commercial real-time middleware with different distribution models, including RT-CORBA and the DSA.

Finally, there is also some research about the specific topic of Ravenscar compliant distributed systems. On the one hand, some middlewares are implemented using the

* This work has been funded in part by the Spanish Ministry of Science and Technology (MCYT), project TICTIC2005-08665-C03-01 (THREAD), and by the Council for Education of the Community of Madrid and the European Social Fund.

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

Ravenscar profile —PolyORB, mentioned above, and DEAR-COTS [18], a distribution framework mainly designed for fault tolerance— and on the other hand work by Audsley and Wellings [4] in IRTAW 2000 about using the DSA and Ravenscar for High-Integrity Systems.

3 Selection of a distribution mechanism

Different distribution mechanisms for Ada are currently used in industry, some of them for specific environments (e.g. SOIS [19] for the space domain, or MPI [23] for scientific computations) while others are multi-purpose, like the Real-Time Common Object Request Broker Architecture (Real-Time CORBA), the Data Distribution Service (DDS), or the Distribute Systems Annex (DSA). The standard distribution mechanism of Ada has different advantages and disadvantages with respect to other middlewares.

3.1 DSA advantages

The Distributed Systems Annex is integrated in the language since Ada 95, and was designed for a wide variety of distributed systems. It is easy to learn [11], and a full distributed application can be developed very quickly adding only some categorization pragmas to specific library units, and specifying the location of each partition using the configuration file. Automatic consistency checks at the start of the application ensure the binary compatibility of each partition [9,§E.3(6)].

Every distributed application is also a valid centralized program, therefore the same application can be first compiled and tested as a single binary and later the exactly same source code can be compiled as a distributed program [9, §E(7)]. This is very useful during the development stage, because debugging a distributed application is far more difficult than a centralized one. Although this can also be achieved using other distribution mechanisms it would require the modification of the source code (and thus bugs can be introduced in the process).

Other advantages of the distribution model (categorization pragmas versus the API of other middlewares) are the compiler checks among partitions for code verification and optimization [9, § E(8)], and for external tools like the SPARK examiner[5], which does not need to be modified to analyze the application. It should be investigated whether the DSA facilitates writing a tool to obtain the temporal model of a system from the source code.

The DSA provides multiple communication paradigms including message passing, distributed shared memory [12], remote procedure calls, and distributed objects. The Ada specification specifies a standard communication subsystem (System.RPC), but implementations are free to generate calling stubs that leverage other underlying middlewares [9, § E.5(27.b/2)] (e.g. DSA implementations exits for CORBA, web services [25] or Java RMI [17]). Each partition has an independent run-time system thanks to the purity rules enforced by the categorization pragmas.

3.2 DSA disadvantages

However, the DSA has also some problems. It was not designed for real-time communication, in opposition with DDS, RT-CORBA, or SOIS. It has also limited vendor support, and there are only a few research and commercial implementations, while CORBA or SOIS have even implementations for safety-critical software.

Although the Ada 95 language designers created a modular scheme to encourage that the compiler and communication subsystem were created by different vendors, in practice there is no interoperability among the implementations. And consistency checks make difficult the implementation of open systems and dynamic services.

Lik ein other distribution mechanisms (e.g. JavaRMI), remote tagged objects cannot migrate to other nodes (they must be limited) so remote dispatching operations are always served by the node that created such tagged object. However, in RMI there is a default serialization for all types, but in the DSA the application programmer must provide the code for the marshalling and unmarshalling of data involving some access types—e.g. a linked list.

The purity rules are very strict, and only preelaborated types can be transferred. For example, no standard time type can be used in a remote operation, the application programmer is forced to define one. In practice, purity rules difficult the reuse of code not designed for adistributed application. However, the same applies to the rest of distribution mechanisms.

Finally, the DSAdoes not provide the Publish/Subscribe [20] communication paradigm, in opposition with DDS. This is an efficient real-time communication paradigm that allows multicast communications, and thus it is very interesting for control systems (especially since the schedulability analysis of multi-event synchronization can be achieved [8]).

3.3 Distribution mechanism chosen

Some of the above disadvantages are not intrinsic to the definition of the DSA but implementation dependent, like interoperability or real-time communication support. The ARM allows the addition of new rules, categorization pragmas, or interfaces of new distribution services. But it is worth noting that if those services are not standarized vendors have little motivation to implement them.

However, some of those disadvantages are not a problem for a HIS. For example, although CORBA offers better interoperability among different vendors this is not usually a core concern in a HIS because the developers have more control over the whole software stack. Although complex distributed high-integrity systems are usually developed by more than one contractor, each safety-critical subsystem is made by only one.

Reuse of past code written in different languages is also desirable, but for HISs the testing and certification steps are more costly. As said above, the DSA simplifies testing because exactly the same source code can be tested as a distributed or centralized program, and certification is

easier thanks to the whole application checks and because less code is written due to its higher abstraction level compared to the others communication mechanism (on the other hand, if more code is generated the programmer has less control of the code, so this should be further investigated).

Some DSA problems found in Ada 95 has been fixed in the technical corrigendum 1 (like some defects related with heterogeneous systems [2]) and in Ada 2005 (more implementation freedom because it is not required to use the partition communication subsystem interface defined in System.RPC). In summary, the DSA is a good foundation for the development of distributed systems, namely HIS ones. In the next section we will discuss the specific problems that must be fixed in the DSA to be able to develop a hard real-time distributed HIS.

4 Restrictions

The Annex E is not disallowed in the Ravenscar profile as defined in Ada 2005. Moreover, at least one of the DSA implementations —PolyORB— can be configured to be Ravenscar compliant. However, the language standard is not designed for distributed real-time systems, so a set of changes is needed to adapt the Annex E to safety-critical distributed systems with hard deadlines.

Some features needed for basic real-time communication were proposed in the past, like messages priorities [14] or non-blocking asynchronous RPCs [7]. These were successfully implemented in RT-GLADE [13], a research real-time DSA prototype. In the future, another desirable addition would be Publish/Subscribe, but the compatibility with the current distribution model should be further investigated.

But for high-integrity systems additional restrictions are needed if the code must be certified, and also to predict if all deadlines will be met. We will restrict the DSA according to current response time analysis theory for distributed systems. The objective is to introduce the minimum restrictions to retain the required programming expressiveness and to permit the reuse of existing code.

4.1 Compulsory restrictions

The restrictions were designed to obtain the required degree of predictability and timeliness needed in a hard real-time system, and to simplify the implementation of a safety-critical middleware, thus easing its certification. In some cases, a restriction could also be introduced to increase performance. And because implementers do not want to maintain more than one specialized run-time, the restrictions must also be compatible with the existing Ravenscar profile. This is the set of restrictions proposed:

- **No remote access types**: A connection for each remote operation is created at elaboration time between the sender and receiver partitions. It is never closed, and new connections should not be established after elaboration time to avoid an excessive blocking time (e.g. similar to the No_Local_Protected_Objects, or No_Task_Allocators restrictions). Therefore, the exact number of interconnections must be known at compile time (actually when the distributed application is configured [9, §E.1(15)], just before the binary for each partition is generated). This can be hard to predict if remote-access-types (access types to remote subprograms and to remote class-wide-types) are allowed. Another advantage is that the set of resources needed for all connections can be predicted at compile time.
- **No concurrent remote calls**: For predictability reasons, a remote operation cannot be called while processing a past invocation (to the same remote subprogram). This implies there is no wait queue in every remote operation, easing the response time analysis. This is similar to the existing restriction Max_Entry_Queue_Length => 1. Another advantage derived from this rule is the absence of loops in the call graph, so no distributed deadlock can occur [21] if there is one RPC handler for each operation of the interface (static allocation of incoming operations, i.e. there is no thread pool). Violations to this rule cannot be detected by the compiler, only at run-time or with tools similar to the SPARK examiner.
- **Coordinated elaboration**: A distributed application cannot start until all its partitions had been elaborated. The DSA is designed for general purpose distributed systems, where some partitions can start executing before others, for example in a client-server scheme [9, § E.1(13)]. However, in a real-time system it is not acceptable to enqueue a remote call until the invoked partition completes its elaboration [9, § E.4(14)]. It also implies that there is a static number of partitions at compilation time. This restriction is similar to the sequential elaboration policy specified by the Ada 2005 pragma Partition_Elaboration_Policy, but it should be noted that this pragma (from Annex H) is not required by the Ravenscar profile.

One implication of the first restriction is that it would disallow the distributed object paradigm. Other concern about no dynamic connections is how to achieve fault tolerance, one of the potential advantages of a distributed system. Transparent partition replication can be the answer (integrating a framework for fault tolerance directly in the DSA implementation instead of in each application [18]), but this issue must be further investigated.

The second restriction is needed for predictability reasons, as done to protected objects when removing the entry queue. And if only one task can be associated with a protected entry, the same also applies with each remote subprogram. Note that local deadlocks where avoided in Ravenscar thanks to the Priority Ceiling Protocol, so the absence of all types of deadlocks is an interesting property for High-Integrity Systems.

It should be noticed that partition termination is already disallowed: In the full DSA a partition can terminate either because all their tasks have finished or when its environment task is aborted. But this cannot happen in this

high-integrity DSA because both restrictions are already enforced by the Ravenscar profile (No_Task_Termination and No_Abort_Statements). No partition termination implies that a remote subprogram call cannot be cancelled [9, §E.4(13)], simplifying the implementation of the middleware.

4.2 Optional restrictions

Other restrictions were considered but later discarded because —although useful for some types of high-integrity systems— are not essential to perform a response time analysis of the system or to simplify the implementation of the run-time:

- *Synchronous calls*: if only asynchronous communication is allowed then the RTA is simplified, but this will reduce the programming expressiveness. Audsley [4] proposed to disallow synchronous calls to avoid excessive blocking time, but newer RTA methods reduce the pessimism introduced in that situations [16]. However, only asynchronous operations should be used if the distributed application is implemented in SPARK. Otherwise, an exception can be raised if there is a communication error while performing a synchronous call.
- *Nested calls*: if a synchronous remote subprogram cannot perform another (blocking) remote call before returning to the caller (a chain of calls) the response-time analysis is greatly simplified. However, the programming expressiveness will be also greatly reduced, and although no nested calls is a sufficient condition to avoid distributed deadlocks, they are already avoided if concurrent calls are disallowed.
- *Complex interpartition communication*: if unconstrained types or complex data structures (e.g. linked list) are used as parameters in a remote operation, it could be impossible to calculate the size of the maximum message. But these types (if correctly used) do not necessarily introduce any schedulability problems, and the programming expressiveness would be greatly reduced if this restriction is introduced.

Ravenscar deals with concurrent code, and disallowing the transmission of these types would be equivalent to restrict a sequential construct. The application programmer should be allowed to have dynamic size messages, but depending on the level of certification required these types can be disallowed with the aid of external ASIS tools.

It is worth noting that the programmer must provide the adequate marshalling and unmarshalling code ('Write and 'Read attributes) for types composed by *non-remote* access types (see example 1). Therefore the programmer is aware of the serialization costs, and the run-time does not have to handle the serialization of complex data (like recursive types [22]). Note that remote access to wide-access types has no serialization problems because they must be limited. They are not disallowed due to serialization costs but to avoid dynamic connections.

4.3 Supported features

To summarize, thedistribution features supported are:

- **Passive partitions**: shared passive or pure packages including atomic and volatile variables, and protected objects (without entries).
- **Static remote subprogram calls**: remote type packages as restricted above, and remote call interface library units.
- **Unconstrained parameters**: unconstrained types and (non-remote) access types are allowed in remote calls.
- **Synchronous and asynchronous communication**: Synchronous communication for active and passive partitions, and pragma Asynchronous to enable asynchronous communication. Pragma All_Calls_Remote is also allowed, useful mainly for code debugging.

5 Implementation requirements

It is desirable that a task invoking a remote operation does not delegate the message generation (including data marshalling, message partitioning, composition of message headers, and even message queueing) to another task to avoid priority inversion. The network is usually non preemptable, so total priority inversion is in general not possible but it can be bounded. The receiver should then process each call with thepriority specifiedin the message.

Each remote operation should be processed in the called partition by a specific thread, including each instantiation of a generic remote call interface (probably each generic instantiation will be in different active partitions, but when more than one is located in the same partition a common thread for each operation is not allowed). Therefore, the ARM requirement to have a reentrant RPC handler is no longer needed in this restricted DSA.

The Program_Error exception is sent back to the caller in the case the destination thread is still processing another call, as done in the Ravenscar profile for tasks trying to access an entry of a protected object in which another task is already waiting. This cannot be detected at the calling partition, only when the calling petition arrives to the server, wasting some bandwidth. However, this should only happen in the testing phase because it implies that the program is erroneous.

It should be noticed that each partition can still have an independent run-time system. No clock synchronization is needed because the communication is message oriented [15, p. 1.27], but of course a mechanism to obtain a certain degree of common time is desirable in a real-time system. This should be further investigated.

The implementation must document the communication process, specifying if any step is delegated by another task in the caller or called partition. It must be further investigated the metrics that should be documented by the implementation.

6 Examples

6.1 Fault-Tolerant inter-node communication

In the first use case the Flight Management System communicates with different nodes, either using a high-speed network, or through a bus (when the bandwidth requirements arelow). The communication links are replicated for fault-tolerance: To recover from transmission errors, and to tolerate a broken link.

In the code of example 1 the Flight Management System communicates with the Flight Plan Manager and with the fuel-level sensor. The fuel-level sensor has low bandwidth and CPU requirements, executing over a microcontroller (minimal run-time system, notasking). The sensors are also replicated. The Flight Plan Manager sometimes has high-bandwidth requirements because in this example it must transfer the complete planned route as a linked list to the Flight Management System.

In our fictitious DSA implementation the middleware handles transparently the replicated networks. But it should be noticed that the replicated sensors are managed by specific application code and not by the DSA implementation.

6.2 Criticality segregation

Usually, the software of a high-integrity system has different criticality levels. For example, Level A code is considered mission critical, while Level B code will not lead to catastrophic events if the software fails. A lower criticality level implies less certification requirements, and thus different verification and validation costs.

In this use case two hypothetical applications of different criticality levels execute in the same hardware node, a common approach in Integrated Modular Avionics (IMA). The RTOS provides a different memory space and CPU budget for each one, and a shared memory region for communication between them. In the DSA terminology, each application is an active partition, while the shared memory region is a passive partition [9, §E.1(2)].

As can be seen in the example 2, the Flight Management System writes the telemetry data in the passive partition (pragma Shared Passive), while a task of the lower criticality partition updates the displays. In our hypothetical DSA implementation the Level A partition can be configured to have R/W access to the passive partition and read access only to the Level B partition. Therefore both applications are completely isolated (so they can be certified at different criticality levels) while the communication is very fast and completely asynchronous.

7 Conclusions and future work

This position paper has discussed the changes needed in the Ada Distributed Systems Annex (DSA) for developing safety-critical distributed systems. Although currently the DSA cannot be used in a distributed system with hard real-time communication requirements, itis argued that the Annex E is more adequate for this kind of High-Integrity Systems than other industrial middlewares.

This paper briefly describes some of the real-time capabilities needed for basic real-time communication, and restricts the DSA to enable certification and to obtain the required degree of predictability and timeliness. The resulting profile is compatible with Ravenscar, and it is believed it has enough programming expressiveness for the development of complex safety-critical hard real-time embedded distributed systems.

Finally, some topics like the documentation requirements and fault-tolerance (probably through transparent replication) must be further investigated. And a prototype is needed in the future to validate the proposed distribution mechanism, and to prove whether it can be successfully implemented and certified.

References

[2] Ada Rapporteur Group. *Ada Issue 249 — Ravenscar profile for high-integrity systems*. Ada Letters, XXV(3), September 2005.

[3] *Ada Issue 208 — What is the meaning of "samerepresentation" in all partitions?*, August 1999.

[4] L. Asplund, B. Johnson, and K. Lundqvist. *Session summary: The Ravenscar profile and implementation issues.* Ada Letters, XIX(25):12–14, 1999. Proceedings of the 9th International Real-Time Ada Workshop.

[5] N. Audsley and A. Wellings. *Issues with using Ravenscar and the Ada distributed systems annex for high-integrity systems.* In IRTAW '00: Proceedings of the 10th international workshop on Real-time Ada workshop, pages 33–39, New York, NY, USA, 2001. ACM Press.

[6] J. Barnes. *High Integrity Software: The SPARK Approach to Safety and Security*. Addison Wesley, 2003.

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

[8] J. J.Gutiérrez and M. González Harbour. *Towards a real-time Distributed System Annex in Ada.* Ada Letters, XXI(1), 2001.

[9] J. J.Gutiérrez, J. Palencia, and M. González Harbour. *Schedulability analysis of distributed hard real-time systems with multiple-event synchronization*. In Proc. 12th Euromicro Conference on Real-Time Systems, pages 15–24. IEEE CS Press, June 2000.

[10] ISO SC22/WG9. *Ada 2005 Annotated Reference Manual.* ISO/IEC 8652:1995(E) with Technical Corrigendum 1 and Amendment 1, 2006. Available on http://www.adaic. com/standards/ada05.html.

[11] M. Joseph and P. Pandya. *Finding response times in real-time systems*. BCS Computer Journal, 29(5):390–395, 1986.

[12] Y. Kermarrec. *CORBA vs. Ada 95 DSA: a programmer's view.* In SIGAda'99: Proceedings of the

1999 annual ACM SIGAda international conference on Ada, pages39–46, New York, NY, USA, October 1999. ACM Press.

[13] P. Ledru and S. G. Shiva. *Interpartition communication with shared active packages.* In TRI-Ada '96: Proceedings of the conference on TRI-Ada '96, pages 57–62, New York, NY, USA, 1996. ACM Press.

[14] J. López Campos, J. J. Gutiérrez, and M. González Harbour. *The chance for Ada to support distribution and real-time in embedded systems*. In A. Llamosí and A. Strohmeier, editors, 9th International Conference on Reliable Software Technologies — Ada-Europe 2004, number 3063 in LNCS, pages 91–105, Palma de Mallorca (Spain), 2004. Springer-Verlag.

[15] S. A. Moody. *Session summary: Distributed Ada and real-time*. In IRTAW '99: Proceedings of the ninth international workshop on Real-time Ada, pages 15–18, NewYork, NY, USA, March 1999. ACM Press. Chairman: Michael González Harbour. Rapporteur: Scott Arthur Moody.

[16] J. C. Palencia Gutiérrez, *Análisis de planificabilidad de Sistemas Distribuidos de Tiempo Real basados en prioridades fijas.* PhD thesis, Universidad de Cantabria, 1999. Supervisor: Michael González Harbour.

[17] J. C. Palencia Gutiérrez, and M. González Harbour. *Exploiting precedence relations in the schedulability analysis of distributed real-time systems.* In RTSS 1999: Proceedings of the 20th IEEE Real-Time Systems Symposium, pages 328– 339, December 1999.

[18] L. Pautet and S. Tardieu. *What future for the distributed systems annex?* In SIGAda '99: Proceedings of the 1999 annual ACM SIGAda international conference on Ada, pages 77–82, NewYork, NY, USA, 1999. ACM Press.

[19] L. M. Pinho and F. Vasques. *Using Ravenscar to support fault tolerant real-time applications*. Ada Letters, XXII(4):47–52,2002.

[20] C. Plummer and P. Plancke. *The spacecraft onboard interfaces, SOIS, standardisation activity*. In DASIA 2002 — Data Systems in Aerospace, 2002.

[21] R. Rajkumar, M. Gagliardi, and L. Sha. *The real-time publisher/subscriber inter-process communication model for distributed real-time systems: design and implementation.* In Proceedings of the First IEEE Real-Time Technology and Applications Symposium (RTAS'95), 1995, pages 66–75, Los Alamitos, CA, USA, May 1995. IEEE Computer Society.

[22] C. Sánchez, H. B. Sipma, Z. Manna, V. Subramonian, and C. Gill. *On efficient distributed deadlock avoidance for real-time and embedded systems*. In Proceedings of the 20th International Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. IEEE Computer Society, April 2006.

[23] D. Tejera, A. Alonso, and M. A. de Miguel. *Predictable serialization in Java*. In IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'07), May 2007.

[24] K. Thomas. *Parallel programming in Ada 95 and MPI*. Ada User Journal, 21(2):143–152, July 2000.

[25] K. Tindell and J. Clark. *Holistic schedulability analysis for distributed hard real-time systems.* Microprocessing and Microprogramming, 40(2-3):117–134, April 1994. Euromicro Journal (Special Issue on Parallel Embedded Real-Time Systems).

[26] 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.

Example 1. Internode communication. Of course, a Remote Call Interface is assigned only to one active partition. A Remote Types unit is needed in this example because access types cannot be declared in a Remote Call Interface.

```
-- Node 1
package Sensors.Fuel is
pragma Remote_Call_Interface;
  type Fuel_Level is delta 0.001 range 0.0 .. 100.0;
  function Current_Fuel_Level return Fuel_Level ;
end ;

-- A Remote Types unit is replicated in every partition
   that includes it (node 2 and 3).
with GPS;
with Ada. Streams;
package Routing is
  pragma Remote_Types;

  type Flight_Plan is private;
  procedure Flight_Plan_Write (
                Stream : not null access
                    Ada.Streams.Root_Stream_Type'Class;
                Item : in Flight_Plan);
  procedure Flight_Plan_Read (
                Stream : not null access
                    Ada.Streams.Root_Stream_Type'Class;
                Item : out Flight_Plan );
  for Flight_Plan'Write
      use Flight_Plan_Write;  - - user-defined marshalling
  for Flight_Plan'Read
      use Flight_Plan_Read ; - - user-defined unmarshalling
private
  type Flight_Plan is record -- Linked list
    Waypoint : GPS.Coordinates;
    Next : access Flight_Plan;
  end record;
end;

-- Node 2
with Routing ;
package Flight_Plan_Management is
  pragma Remote_Call_Interface;
  procedure Planned_Route (
                  Route: out Routing.Flight_Plan);
end;

-- Node 3
with Sensors.Fuel;
with Routing;
with Flight_Plan_Management;
procedure Flight_Management_System is
  Fuel : Sensors.Fuel.Fuel_Level;
  Route : Routing.Flight_Plan;
  -- ...
begin
  loop
    -- ...
    Fuel := Sensors.Fuel.Current_Fuel_Level;
    -- ...
    Flight_Plan_Management.Planned_Route (Route);
    -- ...
  end loop;
end;
```

Example 2. Criticality segregation. A Shared Passive unit can be assigned only to one partition because it has state and can declare public variables. It is preelaborated, and can depend only on Pure units or other Shared Passive packages.

```
-- Shared memory area partitions Level A and B
with Instruments;
package Telemetry is
  pragma Shared_Passive;

  Current_Altitude    : Instruments.Altitude;
  Current_Latitude    : Instruments.Latitude;
  Current_Longitude : Instruments.Longitude;
  Current_TAS         : Instruments.True_Airspeed;
  pragma Atomic (Current_Altitude);
  pragma Atomic (Current_Latitude);
  pragma Atomic (Current_Longitude);
  pragma Atomic ( Current_TAS );
end;

-- Partition Level B
with Telemetry;
package body Displays is

  task body Display_Manager is
    -- ...
  begin
    loop
      -- ...
      Print_Display1 (  Telemetry.Current_Altitude,
                        Telemetry.Current_Latitude,
                        Telemetry.Current_Longitude,
                        Telemetry.Current_TAS );
      -- ...
    end loop ;
  end;
end;

-- Partition Level A
with Telemetry;
with Sensors. Altitude;
procedure Flight_Management_System is
  -- ...
begin
  loop
    -- ...
    Telemetry.Current_Altitude :=
              Sensors.Altitude.Current_Altitude;
    -- ...
  end loop;
end;
```

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 #1: Limited Types in Ada 2005 — Limited Aggregates

Bob Duff, AdaCore

Date: 14 May 2007

Abstract: Ada 2005 allows construction of limited objects via aggregates, thus gaining the advantage of the full coverage rules, which was previously available only for non limited types.

Let's get started…

One of my favorite features of Ada is the "full coverage rules" for aggregates. For example, suppose we have a record type:

```
type Person is
  record
    Name : Unbounded_String;
    Age : Years;
  end record;
```

We can create an object of the type using an aggregate:

```
X : constant Person :=
  (Name => To_Unbounded_String (”John Doe”),
   Age => 25);
```

The full coverage rules say that every component of Person must be accounted for in the aggregate. If we later modify type Person by adding a component:

```
type Person is
  record
    Name : Unbounded_String;
    Age : Natural;
    Shoe_Size : Positive;
  end record;
```

and we forget to modify X accordingly, the compiler will remind us. Case statements also have full coverage rules, which serve a similar purpose.

Of course, we can defeat the full coverage rules by using "others" (usually for array aggregates and case statements, but occasionally useful for record aggregates):

```
X : constant Person :=
  (Name => To_Unbounded_String (”John Doe”),
   others => 25);
```

According to the Ada RM "others" here means precisely the same thing as "Age | Shoe_Size". But that's wrong: what "others" really means is "all the other components, including the ones we might add next week or next year". That means you shouldn't use "others" unless you're pretty sure it should apply to all the cases that haven't been invented yet.

So far, this is old news — the full coverage rules have been aiding maintenance since Ada 83. So what does this have to do with Ada 2005?

Suppose we have a limited type:

```
type Limited_Person is limited
  record
    Self : Limited_Person_Access :=
             Limited_Person’Unchecked_Access;
    Name : Unbounded_String;
    Age : Natural;
    Shoe_Size : Positive;
  end record;
```

This type has a self-reference; it doesn't make sense to copy objects, because Self would end up pointing to the wrong place. Therefore, we would like to make the type limited, to prevent programmers from accidentally making copies. After all, the type is probably private, so the client programmer might not be aware of the problem. We could also solve that problem with controlled types, but controlled types are expensive, and add unnecessary complexity if not needed.

In Ada 95, aggregates were illegal for limited types. Therefore, we would be faced with a difficult choice: Make the type limited, and initialize it like this:

```
X : Limited_Person;
X.Name := To_Unbounded_String ("John Doe");
X.Age := 25;
```

which has the maintenance problem the full coverage rules are supposed to prevent. Or, make the type nonlimited, and gain the benefits of aggregates, but lose the ability to prevent copies.

In Ada 2005, an aggregate is allowed to be limited; we can say:

```
X : aliased Limited_Person :=
  (Self => null, - - Wrong!

   Name => To_Unbounded_String (”John Doe”),
   Age => 25,
   Shoe_Size => 10);
X.Self := X’Access;
```

We'll see what to do about that "Self => **null**" in a future gem.

One very important requirement should be noted: the implementation is required to build the value of X "in place"; it cannot construct the aggregate in a temporary variable and then copy it into X, because that would violate the whole point of limited objects — you can't copy them

Ada Gem #2: Limited Types in Ada 2005 — <> Notation in Aggregates

Bob Duff, AdaCore

Date: 21 May 2007

Abstract: The <> notation may be used in aggregates to request the default value for certain components.

Let's get started…

Last week [previous gem in this issue —ed], we noted that Ada 2005 allows aggregates for limited types. Such an aggregate must be used to initialize some object (which includes parameter passing, where we are initializing the formal parameter). Limited aggregates are "built in place" in the object being initialized.

Here's the example:

```
type Limited_Person is limited
  record
    Self : Limited_Person_Access :=
              Limited_Person'Unchecked_Access;
    Name : Unbounded_String;
    Age : Natural;
    Shoe_Size : Positive;
  end record;
X : aliased Limited_Person :=
  (Self => null, - - Wrong!

   Name => To_Unbounded_String ("John Doe"),
   Age => 25,
   Shoe_Size => 10);
X.Self := X'Access;
```

It seems uncomfortable to set the value of Self to the wrong value (**null**) and then correct it. It also seems annoying that we have a (correct) default value for Self, but in Ada 95, we can't use defaults with aggregates. Ada 2005 adds a new syntax in aggregates — "<>" means "use the default value, if any".

Here, we can say:

```
X : aliased Limited_Person :=
  (Self => <>,
   Name => To_Unbounded_String ("John Doe"),
   Age => 25,
   Shoe_Size => 10);
```

The "Self => <>" means use the default value of Limited_Person'Unchecked_Access. Since Limited_Person appears inside the type declaration, it refers to the "current instance" of the type, which in this case is X. Thus, we are setting X.Self to be X'Unchecked_Access.

Note that using "<>" in an aggregate can be dangerous, because it can leave some components uninitialized. "<>" means "use the default value". If the type of a component is scalar, and there is no record-component default, then there is no default value.

For example, if we have an aggregate of type String, like this:

```
Uninitialized_String_Const : constant String :=
            (1..10 => <>);
```

we end up with a 10-character string all of whose characters are invalid values. Note that this is no more nor less dangerous than this:

```
Uninitialized_String_Var : String (1..10); -- no initialization
Uninitialized_String_Const : constant String :=
      Uninitialized_String_Var;
```

As always, one must be careful about uninitialized scalar objects.

Ada Gem #12: Limited Types in Ada 2005 — <> Notation Part 2

Bob Duff, AdaCore

Date: 8 October 2007

Abstract: We show how the <> notation in aggregates may be used to make better use of record-component default values, thus avoiding duplication of code.

Let's get started…

Have you ever written Ada 95 code like this?

```
package P is
  type T is private;
  …
private
  type T is
    record
      Color : Color_Enum := Red;
      Is_Gnarly : Boolean := False;
      Count : Natural;
    end record;
end P;
package body P is
  Object_100 : constant T :=
    (Color => Red, Is_Gnarly => False, Count => 100);
  …
end P;
```

We want Object_100 to be a default-initialized T, with Count equal to 100. It's a little bit annoying that we had to write the default values Red and False twice. What if we change our mind about Red, and forget to change it in all the relevant places?

The "<>" notation comes to the rescue. If we want to say, "make Count equal 100, but initialize Color and Is_Gnarly to their defaults", we can do this:

```
Object_100 : constant T :=
  (Color => <>, Is_Gnarly => <>, Count => 100);
```

On the other hand, if we want to say, "make Count equal 100, but initialize all other components, including the ones we might add next week, to their defaults", we can do this:

```
Object_100 : constant T := (Count => 100, others => <>);
```

Note that if we add a component "Glorp : Integer;" to type T, then the "others" case leaves Glorp undefined just as this Ada 95 code would do:

```
Object_100 : T;
Object_100.Count := 100;
```

Think twice before using "others".

Ada Gem #3: Limited Types in Ada 2005 — Constructor Functions

Bob Duff, AdaCore

Date: 28 May 2007

Abstract: Constructor functions may be used to create objects of limited types. The result of such function calls is built "in place" in the actual object being created.

Let's get started…

Given that Ada 2005 allows build-in-place aggregates for limited types, the obvious next step is to allow such aggregates to be wrapped in an abstraction — namely, to return them from functions. After all, interesting types are usually private, and we need some way for clients to create and initialize objects.

```
package P is
   type T (<>) is limited private;
   function Make_T (Name : String)
            return T; - - constructor function
private
   type T is limited
      record
         Name : Unbounded_String;
         My_Task : Some_Task_Type;
         My_Prot : Some_Protected_Type;
      end record;
end P;
package body P is
   function Make_T (Name : String) return T is
   begin
      return (Name => To_Unbounded_String (Name),
              others => <>);
   end Make_T;
end P;
```

In Ada 95, constructor functions (that is, functions that create new objects and return them) are not allowed for limited types. Ada 2005 allows fully-general constructor functions. Given the above, clients can say:

```
My_T : T := Make_T (Name => "Bartholomew Cubbins");
```

As for aggregates, the result of Make_T is built in place (that is, in My_T), rather than being created and then copied into My_T. Adding another level of function call, we can do:

```
function Make_Rumplestiltskin return T is
begin
   return Make_T (Name => "Rumplestiltskin");
end Make_Rumplestiltskin;
Rumplestiltskin_Is_My_Name : constant T :=
      Make_Rumplestiltskin;
```

It might help to understand the implementation model: In this case, Rumplestiltskin_Is_My_Name is allocated in the usual way (on the stack, presuming it is declared local to some subprogram). Its address is passed as an extra implicit parameter to Make_Rumplestiltskin, which then passes that same address on to Make_T, which then builds the aggregate in place at that address. Limited objects must never be copied! In this case, Make_T will initialize the Name component, and create the My_Task and My_Prot components, all directly in Rumplestiltskin_Is_My_Name.

Note that Rumplestiltskin_Is_My_Name is constant. In Ada 95, it is impossible to create a constant limited object, because there is no way to initialize it.

As in Ada 95, the "(<>)" on type T means that it has "unknown discriminants" from the point of view of the client. This is a trick that prevents clients from creating default-initialized objects (that is, "X : T;" is illegal). Thus clients must call Make_T whenever an object of type T is created, giving package P full control over initialization of objects.

Ideally, limited and nonlimited types should be just the same, except for the essential difference: you can't copy limited objects. Allowing functions and aggregates for limited types in Ada 2005 brings us very close to this goal. Some languages have a specific feature called "constructor". In Ada, a "constructor" is just a function that creates a new object. Except that in Ada 95, that only works for nonlimited types. For limited types, the only way to "construct" on declaration is via default values, which limits you to one constructor. And the only way to pass parameters to that construction is via discriminants. In Ada 2005, we can say:

```
This_Set : Set := Empty_Set;
That_Set : Set := Singleton_Set (Element => 42);
```

whether or not Set is limited. "This_Set : Set := Empty_Set;" seems clearer to me than:

```
This_Set : Set;
```

which might mean "default-initialize to the empty set" or might mean "leave it uninitialized, and we'll initialize it in later".

Ada Gem #11: Limited Types in Ada 2005 — Constructor Functions Part 2

Bob Duff, AdaCore

Date: 1 October 2007

Abstract: We show here how limited constructor functions can be used in various contexts to build new limited objects in place.

Let's get started…

We've earlier seen examples of constructor functions for limited types similar to this:

```
package P is
   type T (<>) is limited private;
   function Make_T (Name : String)
            return T; -- constructor function
private
   type T is new Limited_Controlled with
      record
         …
      end record;
end P;
package body P is
   function Make_T (Name : String) return T is
```

```
  begin
    return (Name => To_Unbounded_String (Name),
            others => <>);
  end Make_T;
end P;
function Make_Rumplestiltskin return T is
begin
   return Make_T (Name => “Rumplestiltskin”);
end Make_Rumplestiltskin;
```

It is useful to consider the various contexts in which these functions may be called. We've already seen things like:

```
Rumplestiltskin_Is_My_Name : T :=
        Make_Rumplestiltskin;
```

in which case the limited object is built directly in a standalone object. This object will be finalized whenever the surrounding scope is left.

We can also do:

```
procedure Do_Something (X : T);

Do_Something (X => Make_Rumplestiltskin);
```

Here, the result of the function is built directly in the formal parameter X of Do_Something. X will be finalized as soon as we return from Do_Something.

We can allocate initialized objects on the heap:

```
type T_Ref is access all T;
Global : T_Ref;

procedure Heap_Alloc is
  Local : T_Ref;
begin
  Local := new T'(Make_Rumplestiltskin);
  if … then
    Global := Local;
  end if;
end Heap_Alloc;
```

The result of the function is built directly in the heap-allocated object, which will be finalized when the scope of T_Ref is left (long after Heap_Alloc returns).

We can create another limited type with a component of type T, and use an aggregate:

```
type Outer_Type is limited
  record
    This : T;
    That : T;
  end record;
Outer_Obj : Outer_Type :=
        (This => Make_Rumplestiltskin,
         That => Make_T (Name => “”));
```

As usual, the function results are built in place, directly in Outer_Obj.This and Outer_Obj.That, with no copying involved.

The one case where we _cannot_ call such constructor functions is in an assignment statement:

```
Rumplestiltskin_Is_My_Name :=
        Make_T(Name => ""); - - Illegal!
```

which is illegal because assignment statements involve copying. Likewise, we can't copy a limited object into some other object:

```
Other : T := Rumplestiltskin_Is_My_Name; - - Illegal!
```

Ada Gem #10: Limited Types in Ada 2005 — Extended Return Statements

Bob Duff, AdaCore

Date: 24 September 2007

Abstract: An extended_return_statement may be used to give a name to the result object being created by a function.

Let's get started…

A common idiom in Ada 95 is to build up a function result in a local object, and then return that object:

```
function Sum (A : Array_Of_Natural) return Natural is
  Result : Natural := 0;
begin
  for Index in A'Range loop
    Result := Result + A (Index);
  end loop;
  return Result;
end Sum;
```

Ada 2005 allows a notation called the extended_return_statement, which allows you to declare the result object and return it as part of one statement. It looks like this:

```
function Sum (A : Array_Of_Natural) return Natural is
begin
  return Result : Natural := 0 do
    for Index in A'Range loop
      Result := Result + A (Index);
    end loop;
  end return;
end Sum;
```

The return statement here creates Result, initializes it to 0, and executes the code between "do" and "end return". When "end return" is reached, Result is automatically returned as the function result.

For most types, this is no big deal — it's just syntactic sugar. But for limited types, this syntax is almost essential:

```
function Make_Task (Val : Integer) return Task_Type is
  Result : Task_Type (Discriminant => Val * 3);
begin
  … - - some statements
  return Result; - - Illegal!
end Make_Task;
```

The return statement here is illegal, because Result is local to Make_Task, and returning it would involve a copy, which makes no sense (which is why task types are limited). In Ada 2005, we can write constructor functions for task types:

```
function Make_Task (Val : Integer) return Task_Type is
begin
   return Result : Task_Type (Discriminant => Val * 3) do
      … - - some statements
   end return;
end Make_Task;
```

If we call it like this:

```
My_Task : Task_Type := Make_Task (Val => 42);
```

Result is created "in place" in My_Task. Result is temporarily considered local to Make_Task during the "… -- some statements" part, but as soon as Make_Task returns, the task becomes more global. Result and My_Task really are one and the same object.

When returning a task from a function, it is activated after the function returns. The "… -- *some statements*" part had better not try to call one of the task's entries, because that would deadlock. That is, the entry call would wait until the task reaches an accept statement, which will never happen, because the task will never be activated.

While the extended_return_statement was added to the language specifically to support limited constructor functions, it comes in handy whenever you want a local name for the function result:

```
function Make_String (…) return String is
   Length : Natural := 10;
begin
   if … then
      Length := 12;
   end if;
   return Result : String (1..Length) do
      … - - fill in the characters
      pragma Assert (Is_Good (Result)); null;
   end return;
end Make_String;
```

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.org