ADA USER JOURNAL

Volume 29
Number 3
September 2008

Contents

Editorial Policy for Ada User Journal

Publication

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

Aims

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

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

The Journal publishes the following types of material:

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

Further details on our approach to these are given below.

Original Papers

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

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

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

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

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

News and Product Announcements

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

Reprinted Articles

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

Commentaries

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

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

Announcements and Reports

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

Reviews

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

Submission Guidelines

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

Editorial

This September issue of the Ada User Journal was indeed the most demanding, and unfortunately also the most delayed, I had so far in my (very) short term as the Journal's Editor. Nevertheless, I am sure that the readers will understand and apologise us for its late appearance, considering the worth of its contents.

First, the issue finalizes the publication of the Proceedings of the 13th International Real-Time Ada Workshop (IRTAW 2007), which started to be published in the Ada User Journal last December, almost one year ago. In the last technical session of the workshop the participants analysed the relation of Ada with other standards, particularly POSIX, as proposed in a paper by Stephen Michell, of Maurya Software, Canada. The last session was dedicated to the analysis of conclusions and open issues coming from the discussions in the workshop, and to future plans for the workshop itself.

Whilst closing the subject of IRTAW 2007, we start to publish material coming from the Ada-Europe 2008 conference. In this issue the reader will find the proceedings of the "Ada and Software Engineering Education" session of the conference. This special session had the objective of discussing the role of programming languages, and of Ada in particular, in the software engineering curriculum, and consisted in the presentation of four position papers and a debate.

In the first paper of these proceedings, the reader will find the report of the session, with a summary of the presentations and the main issues introduced during the debate. Afterwards, the first position paper is by Ed Schonberg and Robert Dewar, from the New York University, USA, providing their analysis on the use of Java as the first programming language, arguing that Ada and C++ are superior for introductory computer science courses. The following paper, by John McCormick from the University of Northern Iowa, USA, presents an interesting experience of students implementing software to control a railroad model and the gains which were obtained by switching from C to Ada. The third position paper is by Jean Pierre Rosen, of Adalog, France, providing an assessment of the importance of education for software engineering in general. Finally, the session closes with a paper by Carl Brandon, of the Vermont Technical College, USA, describing the use of Ada to implement the software for a pico-satellite platform.

We should acknowledge the effort of the organizers and the participants of the session on supporting this concern, which is of particular importance for the future of Ada. There is also a note on the summary of the session on the importance of introducing software reliability in the curricula of computer science and related courses, something that we should all campaign for.

Finally, the Ada Gems section provides two gems on Changing Data Representation, by Robert Dewar. And, as usual, we have the News, Calendar and Forthcoming Events sections, providing the readers with a quick look to the world of Ada.

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

News

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

Contents

Ada-related Organizations

ARA — ACATS 3.0E

From: Ada Information Clearinghouse
Date: July 22, 2008
Subject: Ada Conformity Assessment Test Suite
URL: http://www.adaic.com/whatsnew.html

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

[See also "ARA — ACATS 3.0D" in AUJ 29-2 (Jun 2008), p.77. —su]

Ada-related Events

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

Oct 30 — SIGAda Awards

From: John McCormick <mccormick@cs.uni.edu>
Date: Mon, 18 Aug 2008 10:00:07 −0700 (PDT)
Subject: Call for SIGAda Award Nominations
Newsgroups: comp.lang.ada

Dear Members of the Ada Community:

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

We welcome your nominations of deserving recipients.

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

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

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

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

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

Please note that anyone who has received either of the two awards remains eligible for the other. Perhaps there is an outstanding SIGAda volunteer who has won our Distinguished Service Award and who has also made important contributions to the advance of Ada technology, or 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 SEPTEMBER 21!

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

Thank you,

John McCormick

Chair ACM SIGAda

[See also "Nov 7 — SIGAda Awards" in AUJ 28-4 (Dec 2007), p.201. —su]

Jun 8–12 — Ada-Europe 2009

From: Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be>
Date: Sat, 28 Jun 2008 22:05:49 +0200 (CEST)
Subject: CFP 14th Conf. Reliable Software Technologies, Ada-Europe 2009
Organization: Ada-Europe, c/o Dept. of Computer Science, K.U.Leuven
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc
Summary: Start now to think about your submissions!
Keywords: Conference,tutorials, ind-ustry,reliability,Ada,LNCS,Brest,France

PRELIMINARY CALL FOR PAPERS

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

8 – 12 June 2009, Brest, France

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

Organized by Ada-Europe,

in cooperation with ACM SIGAda (approval pending)

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

General Information

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

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

12 January 2009: Submission of industrial presentation proposals

09 February 2009: Notification to all authors

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

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

08–12 June 2009: Conference

Topics

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

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

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

Call for Regular Papers

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

Proceedings

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

Awards

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

Call for Industrial Presentations

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

Call for Tutorials

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

Call for Workshops

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

Call for Exhibitions

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

Grants for Students

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

[See also "Jun 16–20 — Ada-Europe 2008" in AUJ 29-2 (Jun 2008), p.79. —su]

Ada and Education

Webminar: GNATbench InSight

From: AdaCore Developer Center
Date: Wednesday June 11, 2008
Subject: GNATbench 2.1 InSight webinar
RSS: http://www.adacore.com/2008/06/11/gnatbench-21-insight-webinar-2/

There are still places left for the latest GNAT Pro InSight webinar taking place next Tuesday (June 17). This one will feature GNATbench 2.1.0. This release introduces many new features including project management and presentation enhancements, language-sensitive editor enhancements, additional wizards, builder enhancements, and source code navigation enhancements. This webinar will describe and demo some of the new features introduced in 2.1.0. As always, we will allow a question and answer session at the end enabling you to talk directly with the designers of GNATbench.

This webinar will appeal to Ada developers that are using, or are interested in using, GNAT Pro and the Eclipse development environment in their projects. To register for this event, please visit: www.adacore.com/home/gnatpro/webinars

From: AdaCore Developer Center
Date: Wednesday July 2, 2008
Subject: GNATbench 2.1 InSight webinar available
RSS: http://www.adacore.com/2008/07/02/gnatbench-21-insight-webinar-available/

The recently recorded GNATbench 2.1 InSight webinar is now available for viewing. Please visit www.adacore.com/home/gnatpro/webinars to watch it.

[See also "Webminar: GPS InSight" in AUJ 29-2 (Jun 2007), p.81. —su]

Ada-related Resources

AdaCommons — Ada wiki

From: Jerry <lanceboyle@qwest.net>
Date: Fri, 18 Jul 2008 15:32:09 −0700 (PDT)
Subject: New Ada wiki
Newsgroups: comp.lang.ada

Someone (perhaps someone on this list) has started an Ada wiki.

http://adacommons.org/

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Sat, 19 Jul 2008 11:17:46 +0300
Subject: Re: New Ada wiki
Newsgroups: comp.lang.ada

AdaCommons was started by #Ada IRC channel (at Freenode network) people. If you have access to an IRC client, feel free to join #AdaCommons channel (also at Freenode), which is meant for AdaCommons related discussions.

Or just start editing the wiki and use the talk wiki pages for discussions. To prevent spam, you are needed to login before you can edit pages. (As a bonus you can be relatively anonymous since only your account name is shown.)

[See also "Ada wikibook to be published" in AUJ 28-1 (Mar 2007), p.9. —su]

Source code repositories

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Tue, 1 Jul 2008 05:56:15 +0300
Subject: Re: Learning Ada but missing the basics?
Newsgroups: comp.lang.ada

> Could anyone point me to an SVN repository for a good, well written open source Ada application?

Check listings at

http://www.ohloh.net/projects/search?q=ada&x=0&y=0 ,

http://freshmeat.net/browse/163/ ,

http://sourceforge.net/search/?type_of_search=soft&words=ada , and

http://code.google.com/hosting/search?q=label:Ada

for some repositories. Most of the open source Ada projects are libraries, but if you want an application, you could check AdaControl (http://www.adalog.fr/adacontrol2.htm, no public SVN repository available).

From: Pascal Obry <pascal@obry.net>
Date: Wed, 02 Jul 2008 11:18:48 +0200
Subject: Re: Learning Ada but missing the basics?
Newsgroups: comp.lang.ada

Look at AWS and Templates_Parser code. I have put lot of efforts to be sure the code is clean, documented and readable. It uses many Ada features even some Ada 2005 ones (interfaces, object.method notation, anonymous access…).

https://libre.adacore.com/aws/

If you are adventurous do not hesitate to have a look at the GNAT code itself. It is quite large but really clean.

You'll see that on those projects the code quality and style is uniform in all the sources. This makes a project easier to maintain by a group of people. No one own (by using weird style for example where other developers do not feel comfortable) a part of the code. This to say that there is no ONE good style but most importantly a style must be followed, uniformity is more important to me.

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Tue, 01 Jul 2008 03:13:57 GMT
Subject: Re: Learning Ada but missing the basics?
Newsgroups: comp.lang.ada

But I could point you to the source for the Mine Detector game:

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

[See also "Ada-Europe 2004 Conference" in AUJ 25-2 (Jun 2004), p.43. —su]

Ada-related Tools

Simple components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 29 Jun 2008 21:15:18 +0200
Subject: ANN: Simple Components v3.1
Newsgroups: comp.lang.ada

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

This version is compatible with new GNAT GPL 2008.

Other new things are:

- Lock-free FIFO of indefinite objects;
- For signaled FIFOs waiting for not full or not empty queue can be now canceled from outside;
- A test added for a GNAT GPL 2008 bug.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 27 Jul 2008 11:41:56 +0200
Subject: ANN: Simple components for Ada v3.2
Newsgroups: comp.lang.ada

The current version provides implementations of smart pointers, sets, maps, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support. It grew out of needs and does not pretend to be universal. Tables management and strings editing are described in separate documents see Tables and Strings edit. (…)

New in this version is a portable IEEE 754 floating-point representation support. Generic packages instantiated by an Ada floating-point type are provided for single and double precision IEEE numbers. IEEE NaN, infinities and denormalized numbers are supported.

[See also same topic in AUJ 29-2 (Jun 2008), pp.81–82. —su]

Units of measurement for Ada

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 2 Jul 2008 22:02:01 +0200
Subject: ANN: Units of measurement for Ada v2.7
Newsgroups: comp.lang.ada

The library provides an implementation of dimensioned values for Ada. Unit checks are made at run-time, if not optimized out by the compiler. SI and irregular measurement units are supported. Shifted units like degrees Celsius are supported too. Conversions from and back to strings are provided for all various irregular units. An extensive set of GTK widgets for dealing with dimensioned values is included, though use of GTK is not mandatory for the rest of the library.

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

This version is adapted to recently published GtkAda 2.10.2, which has changed behaviour of popup windows.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Sun, 6 Jul 2008 11:22:16 +0200
Subject: ANN: Units of measurement for Ada v2.8
Newsgroups: comp.lang.ada

(…) The focus of this release is documentation and when necessary adjusting factors of irregular units. When the same unit has multiple definitions, the chosen one is specified. For all irregular unit definition a reference to the source is provided in the documentation.

Also:

- Added irregular units dram, gill, league;
- Added SI derived unit katal (kat);
- Removed irregular unit candle;
- Bq was moved to derived SI units;
- Unicode ounce sign is now supported;
- Bug fix in Get_Value_As.

[See also same topic in AUJ 28-4 (Dec 2007), pp.206–207. —su]

GNAT GPL 2008 Edition

From: Jamie Ayre <ayre@adacore.com>
Date: Thu, 12 Jun 2008 09:25:16 +0200
Subject: GNAT GPL 2008 now available
To: ada-belgium-info@cs.kuleuven.be

We are pleased to announce the release of GNAT GPL 2008, the Ada Toolset for Academic users and FLOSS developers. It introduces hundreds of enhancements including:

- Availability on the Windows .NET platform
- Upgrade of the debugging engine
- Improvement in robustness and efficiency for Ada 2005 features
- Many new warnings & restrictions to help programmers detect errors earlier
- Companion tools such as gprof, gcov, gnatcheck, gnatpp and gnatmetric are being enhanced to support a wider variety of needs, coding styles, and coding standards
- Support for Pre/Post conditions

GNAT GPL 2008 comes with version 4.2.1 of the GNAT Programming Studio IDE and GNATbench 2.1, the GNAT plug-in for Eclipse.

It is available on the GNU Linux (32 and 64 bit), .NET, and Windows platforms.

GNAT GPL 2008 can be downloaded from the “Download GNAT GPL Edition” section on https://libre.adacore.com.

For regular updates on the GNAT technology, please visit the Developer's Center. It includes a developer's log giving updates on the GNAT technology, technical papers, code samples, and documentation.

For more info, please visit:

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

[See also “GNAT GPL 2007 Edition” in AUJ 28-2 (Jun 2007), p.74. —su]

Interval arithmetic

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 7 Jul 2008 22:47:20 +0200
Subject: ANN: Interval arithmetic for Ada v1.6
Newsgroups: comp.lang.ada

The library provides an implementation of intervals for Ada. It includes arithmetic and relational operations.

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

In the new version the dimensioned intervals are based on the version 2.8 of Units of Measurements for Ada; Also a bug was fixed in the Get_Value_As function of dimensioned intervals.

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

GLOBE_3D — 3D Engine

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Sat, 14 Jun 2008 23:48:21 +0200
Subject: Ann: GLOBE_3D release, with collision detection
Newsgroups: comp.lang.ada

…and an automated way of translating game map data from the “id Tech 4” editor (Doom 3, Quake 4). Some screenshots:

http://globe3d.sf.net/g3d/folder1.htm

The main page is: http://globe3d.sf.net/

[See also same topic in AUJ 27-4 (Dec 2006), p.203. —su]

QtAda binding

From: Vadim Godunko <vgodunko@gmail.com>
Date: Thu, 17 Jul 2008 06:30:38 −0700 (PDT)
Subject: Announce: QtAda 1.0.4 released
Newsgroups: comp.lang.ada

We are pleased to announce QtAda 1.0.4 release. This release has supports for the GNAT GPL 2008 and bug fixes.

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

Multiplatform source code package and Microsoft Windows binary package of the QtAda 1.0.4 can be downloaded from:

http://www.qtada.com/

[See also same topic in AUJ 29-2 (Jun 2008), p.83 and “Qt4Ada” in this issue. —su]

Qt4Ada

From: Leonid Dulman <leonid_dulman@yahoo.co.uk>
Date: Sun, 22 Jun 2008 18:04:57 −0000
Subject: Ann. qtada ada interface for Qt4 is availabble.
Newsgroups: comp.lang.ada

Ann. QtAda, an Ada interface for Qt4, is now available. It was tested in Windows and Fedora 9 x86_64 with Qt4.x and GPL GNAT 2008. You may download it from http://www.websamba.com/guibuilder

[See also “QtAda binding” in this issue. —su]

GTKAda contributions

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 30 Jun 2008 21:05:10 +0200
Subject: ANN: GtkAda contributions v2.2
Newsgroups: comp.lang.ada

The library extends GtkAda. It deals with the following issues:

- Tasking support;
- Custom models for tree view widget;

- Custom cell renderers for tree view widget;
- Multi-columned derived model;
- Extension derived model (to add columns to an existing model);
- Abstract caching model for directory-like data;
- Tree view and list view widgets for navigational browsing of abstract caching models;
- File system navigation widgets with wildcard filtering;
- Resource styles;
- Capturing resources of a widget;
- Embeddable images;
- Some missing subprograms and bug fixes;
- Measurement unit selection widget and dialogs;
- Improved hue-luminance-saturation color model;
- Simplified image buttons and buttons customizable by style properties;
- Controlled Ada types for GTK+ strong and weak references;
- Simplified means to create lists of strings.

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

This new version is based on freshly released GNAT GPL 2008 and GtkAda.2.10.2. Other issues:

- In order to improve security Gtk.Persistent_Storage_Browser wipes password strings it creates before their deallocation;
- Columned tree model Gtk.Tree_Model.Columned_Store contains Is_Ancestor and Is_Descendant functions;
- Gtk.Tree_Model.Columned_Store also supports a mode without a reference model;
- The extension tree model Gtk.Tree_Model.Extension_Store supports reference model change;
- The procedure Get_Directory_Cache is added to Gtk.Persistent_Storage_Browser;
- Procedure Set is added to GLib.Object.Weak_References;
- Bug fix in GLib.Object.Strong_References and Weak_References;
- Get_Current_Object, Get_Current_Storage, Get_Path added to Gtk.Persistent_Storage_Browser.

[See also same topic in AUJ 29-1 (Mar 2008), pp.9–10. —su]

GWenerator

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Thu, 14 Aug 2008 13:28:11 −0700 (PDT)
Subject: Ann: GWenerator 0.8
Newsgroups: comp.lang.ada

GWenerator is a code generator that translates a Resource Compiler script file (.rc or .dlg) into an Ada package for the high-level GWindows GUI framework for MS Windows.

Even in absence of a proper interactive GUI builder for GWindows, it allows to use the GUI building capabilities of numerous existing tools (even Visual Studio 2008) and obtain automatically from their output the creation procedures for dialogs and menus in the GWindows "universe".

This version 0.8 is very early but already usable. So far, only the command-line flavour of the tool is available (RC2GW), but anyway both flavours (command-line and graphic) will steer exactly the same code generator.

Feedback is welcome…

GWenerator can be downloaded here: http://sf.net/projects/gnavi

VAD 7.1 for X86-64 — Visual Ada Developer

From: Leonid Dulman <leonid_dulman@yahoo.co.uk>
Date: Sun, 22 Jun 2008 18:00:57 −0000
Subject: Ann VAD 7.1 X86_64 binaries
Newsgroups: comp.lang.ada

Ann. VAD (Visual Ada Developer) 7.1 binaries for X86_64 are available. It was tested in Fedora 9 x86_64 and GPL GNAT 2008 You may download if from http://www.websamba.com/guibuilder

[See also "VAD 7.1 — Visual Ada Developer" in AUJ 25-2 (Jun 2008), p.83. —su]

SOCI-Ada — Database Access Library

From: Maciej Sobczak <maciej@msobczak.com>
Subject: SOCI-Ada: The Database Access Library for Ada
Date: Thu, 7 Aug 2008 02:18:17 −0700 (PDT)
Newsgroups: comp.lang.ada

I am pleased to announce that the first version of SOCI-Ada is available to download:

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

The SOCI-Ada library is built on top of a successful SOCI library that was written for C++ programmers. The advantages of SOCI-Ada are:

- Modular design based on dynamic backend loading. Thanks to this feature, new backends implemented within the context of the main SOCI project are immediately available for Ada programmers without any additional work. A large community of C++ users can help ensure that the new backends are well tested in a variety of environments and usage scenarios.
- Native backends for major database servers ensure optimal performance and minimize configuration overhead and complexity that is usually associated with other database access methods.
- Direct support for bulk operations allow to achieve high performance with queries that operate on large data sets.
- Very liberal open-source license (Boost, accepted by Open Source Initiative) that encourages both commercial and non-commercial use.
- Easy to use and compact interface.

Currently the following backends are available, each communicating with the database server via its native interface (no ODBC required):

- Oracle
- PostgreSQL
- MySQL

The link above is a starting point for both downloads and online docs, but I think the best way to attract attention is to show examples of *full* programs that access a database using this library. Such examples are available here:

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

This is the first version of the library and as such provides only the basic functionality, at least when compared with what is available in the main SOCI project. Your comments are highly welcome and will certainly contribute to my understanding of what further functionality would be needed. Support for BLOB is the most likely candidate for future versions of the library.

All comments welcome.

(BTW — please do not hesitate to contact me in case of any questions related to the main SOCI project)

[See also "ABDI — Ada Unified Database Interface" in AUJ 27-3 (Sep 2006), p.139. —su]

Source code filters

From: Oliver Kellogg <okellogg@users.sourceforge.net>
Date: Sat, 31 May 2008 21:19:31 +0200
Subject: a few Ada related source code filters
Newsgroups: comp.lang.ada

I gathered up some of my perl utilities that accumulated over time and put them here:

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

Right now, there are

- ada2idl.pl helps translate existing Ada types to CORBA IDL
- adareps2c.pl converts Ada record types with representation clauses to equivalent C structs with bit length indications
- obfuscada.pl helps create an obfuscated copy of an entire set of Ada source files, useful for submitting public compiler bug reports on NDA or proprietary code

but one or two more will be added soon.

From: Oliver Kellogg <okellogg@users.sourceforge.net>
Date: Tue, 03 Jun 2008 18:14:33 +0200
Subject: Re: a few Ada related source code filters
Newsgroups: comp.lang.ada

- ada2idl.pl new version 0.2 tidies up the generated code (somewhat)
- adareps2c.pl new version 0.2 adds generation of __attribute__((packed))
- withlist.pl version 0.5 prints all withed units of an Ada unit, including indirectly withed units

From: Oliver Kellogg <okellogg@users.sourceforge.net>
Date: Mon, 09 Jun 2008 06:29:37 +0200
Subject: Re: a few Ada related source code filters
Newsgroups: comp.lang.ada

I added a few more,

- UniLexer.pm is a unified lexer for Pascal- and C-family languages
- indentada.pl indenter for Ada at this point is a feasibility demo for Unilexer.pm but may evolve into a full pretty printer as time permits.

Lex and Yacc alternatives

From: Peter C. Chapin <pcc482719@gmail.com>
Date: Wed, 20 Aug 2008 22:35:51 −0400
Subject: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

I have a need for a parser generator and a lexical analyzer generator that produce Ada. I see that there is an ayacc and aflex project. The main page for it appears to be here:

http://www.ics.uci.edu/~arcadia/Aflex-Ayacc/aflex-ayacc.html

However the download link on that page does not appear to work. It seems like the host no longer exists. Most other references to ayacc point to the same non-existent place. I was able to locate the source code for a slightly older version of ayacc on the TenDRA site. I have not yet located aflex at all.

Is this project completely dead and buried? Is there some other parser generator that I should be looking at instead?

Thanks in advance for any help you might be able to give…

From: J. David Bryan <wqoelna@npz.bet>
Date: Thu, 21 Aug 2008 00:39:59 −0400
Subject: Re: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

Try:

http://www.ics.uci.edu/~self/

There's a working link there.

From: ficorax@gmail.com <ficorax@gmail.com>
Date: Wed, 20 Aug 2008 23:27:40 −0700 (PDT)
Subject: Re: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

I suggest that you should use tool from http://adagoop.martincarlisle.com

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Thu, 21 Aug 2008 10:36:24 +0200
Subject: Re: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

I prefer table-driven approach to generators.

You might find a table-driven parser in simple components:

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

which I am use for the only complex stuff, expressions. In almost any "normal" language the rest is just trivial, easily handled by a recursive descent parser. (I never managed to understand what formal grammars are good for… (:-))

From: Gautier de Montmollin <gdemont@hotmail.com>
Date: Thu, 21 Aug 2008 07:55:07 −0700 (PDT)
Subject: Re: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

If you still need ayacc & aflex, there is an improved version coming with the GWenerator at http://sf.net/projects/gnavi/ Both tools compile "out of the box" with GNAT — and probably any Ada 95+ compiler.

Improvements are:

- added functions to track line & column of parsed code
- possibility of setting table sizes without changing the tools themselves
- the mighty "Syntax Error" indicates line number
- lots of useless "with" or "use" removed in both ayacc & aflex sources, and other details spotted by GNAT

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Thu, 21 Aug 2008 05:56:27 −0400
Subject: Re: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

Depending on what you are really doing, you can use ASIS.

Or use the parser from GNAT.

These don't "generate parsers", but they can be used to write programs that generate Ada code.

From: Colin Paul Gloster <Colin_Paul_Gloster@acm.org>
Date: Fri, 22 Aug 2008 12:02:33 +0100
Subject: Re: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

One mention of ASIS and some mentions of recursive descent parsers have been made. I shall state advice for you in a more direct manner. Do not write anything based on Lex and YACC. They are for bottom-up designs. Learn how to do top-down parsing or otherwise be damned to never write an unbuggy parser.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Fri, 22 Aug 2008 14:56:18 +0200
Subject: Re: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

> Speaking of lexical analysis and recursive descent parsers, I'd like to know whether anyone here has tried OpenToken[1] and would like to comment on it.
>
> [1] http://www.telepath.com/~dennison/Ted/OpenToken/OpenToken.html

I studied it some time ago and found it interesting. In particular, I borrowed the idea of generation of a table from an Ada enumeration type. If a table contains only legal Ada identifiers, it is a very quick way to create it.

To the critique of OpenToken. One problem is that it really concentrates on tokens, nothing else. A recursive descent parser should also support higher-level constructs, like user-defined blank/comment, identifier, literals, and more generally expressions. So that, when you see the token "declare", you could tell the parser get me a name, and then after the colon, the declaration expression etc.

It also pays too little attention to abstracting the sources being parsed and links to the sources (needed for error messages and integration into IDE).

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Fri, 22 Aug 2008 16:29:51 +0300
Subject: Re: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

I use its lexical-analysis functions a lot, for many different purposes. It works quite well enough for me. However, I can generally myself design the language to be analysed. I would not be surprised to find some awkward problems with languages that have been defined by others — but that probably holds for any general lexical-scanner tool.

I very much like the absence of any "generator" step — a simple "gnatmake" is enough to update the application after a change to the token structure.

I have not used the parsing functions in OpenToken; they did not exist when I started to use OpenToken, so I got into the habit of writing my parsers manually.

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Fri, 22 Aug 2008 20:17:46 +0300
Subject: Re: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

> I too like the absence of a generator, and I'm used to writing my own parsers by hand. Did you find that OpenToken helped a lot in doing that?

It does its job: lexical analysis. The parser gets to look at the tokens one by one; OpenToken provides a function to return the identity of the current token (an enumeration), another function to return the text string of the current token, and a procedure to advance to the next token. That is what I expect of a lexical analyser, and OpenToken gives me that (at least; I haven't really studied it thoroughly to see what else there may be).

As far as I recall, the only wart I have found has to do with the error reporting when the input text has a sequence of characters that does not match any token — I had to add a special "invalid token" definition (Opentoken.Recognizer.Nothing.Get) to find the line-number and column-number of the erroneous text. A minor detail.

If I try to think of what might be missing, perhaps the main thing is context-dependent lexical analysis: the ability to say, for example, that I expect the next token to be an identifier, so please ignore the definitions of reserved keywords and just consider them identifiers, too. Of course I have designed my own languages not to need this (keywords are really reserved).

As far as I know OpenToken has no general look-ahead facility, but it should not be too hard to build one yourself, on top of OpenToken, if you need it.

Another good point about OpenToken and the absence of a generator phase: one can have several instances of OpenToken in the same application, for different languages, without any clashes of names or data. My application needs that.

A caveat: The amounts of text that my applications scan with OpenToken are small. I have no idea of the scanning speed; it has been quite enough for my needs.

From: Stephen Leake <stephen_leake@stephe-leake.org>
Date: Sun, 24 Aug 2008 11:02:42 −0400
Subject: Re: Status of ayacc and aflex?
Newsgroups: comp.lang.ada

I use it at work, but not as recursive descent; I write a grammar using it.

I had to fix a couple bugs, and it can be confusing getting the grammar to be unambiguous, but I like the resulting high-level code.

As with other grammar-oriented parsers, the error messages leave a lot to be desired. If I ever find time, I'd like to try switching to using it in a recursive descent way; that usually gives better error messages.

[See also "Lex and Yacc for Ada" in AUJ 25-2 (Jun 2004), p.54. —su]

Cheddar — Real-Time Scheduling Simulator

From: Frank Singhoff <singhoff@univ-brest.fr>
Date: Tue, 1 Jul 2008 17:37:59 +0200
Subject: New release of Cheddar, a GNU GPL real time scheduling analyzer
Newsgroups: comp.lang.ada

We are pleased to announce a new release of Cheddar. Cheddar is a free real time scheduling tool. Cheddar is designed for checking task temporal constraints and buffer sizes of a real time application/system. It can also help you for quick prototyping of real time schedulers. Finally, it can be used for educational purposes. Cheddar is developed and maintained by the LISyC Team, University of Brest. Since 2008, Ellidiss Technologies also contributes to the development of Cheddar and provides industrial support. Cheddar is written in Ada and runs on Linux and win32 boxes.

Thanks to R. Couillet, P. Dissaux, J. Hugues, N. Vienne, P. Wong (bugs fixed/reports)

The current release is now 2.1. If you are a regular Cheddar's user, we strongly advice you to switch to the 2.1 release. See "ChangesLog.pdf" and "FIXED_BUGS.pdf" files for more information.

1) New features summary :

- AADL translation when an AADL file is loaded in Cheddar command line.
- Update Ocarina from 1.0 to 1.1.
- New user-defined scheduler modeling feature : you can now model your scheduler entity synchronizations by a set of timed automata. See the ACM SIGAda Ada Letters article of 2007 for further details.
- Cheddar menu simplification.
- Compilation on Linux and Windows Box simplification.
- Many fixed bugs : important scheduling simulation memory leak fix, precedences graph…

2) General Cheddar features :

Cheddar is composed of two independent parts : an editor used to describe a real time application/system, and a framework. The editor allows you to describe systems composed of several processors which own tasks, shared resources, buffers and which exchange messages. Such a systems specification can be expressed with AADL, the architecture and design language from the SAE. The framework includes many feasibility tests and simulation tools. Feasibility tests can be applied to check that task response times are met and that buffers have bounded size. When feasibility tests can not be applied, the studied application can be analyzed with scheduling and buffer simulations. Cheddar provides a way to quickly define "user-defined schedulers" to model scheduling of ad-hoc applications/systems (ex : ARINC 653). The most important analysis tools are the following :

- Do scheduling simulations with classical real time schedulers (Rate Monotonic, Deadline Monotonic, Least Laxity First, Earliest Deadline First, POSIX queueing policies : SCHED_OTHERS, SCHED_FIFO and SCHED_RR) with different type of tasks (aperiodic, periodic, task activated with a poisson process law, …)
- Extract information from scheduling simulation. (buffer utilization factor, task response times, task missed deadlines, number of preemption, …)
- Apply feasibility tests on tasks and buffers (without scheduling simulation):
  - Compute task response time bounds.
  - Apply processor utilization tests.
  - Compute bound on buffer size (when buffers are shared by periodic tasks)
- Shared resources support (scheduling and blocking time analysis). Supported protocols : PIP, PCP.
- Tools to express and do simulations/feasibility tests with task precedencies :
  - Schedule tasks according to task precedencies
  - Compute Tindell end to end response time.
  - Apply Chetto and Blazewicz algorithms.
- Tools to run scheduling simulation in the case of multiprocessors systems
- Do simulation when tasks are randomly activated.

- Can run scheduling simulation on user-defined scheduler and task arrival patterns.
- Run user-defined analysis on scheduling simulation.
- …

Cheddar is distributed under the GNU GPL license. It's a free software, and you are welcome to redistribute it under certain conditions; See the GNU General Public License for details.

Source code, binaries and documentations can be freely downloaded from http://beru.univ-brest.fr/~singhoff/cheddar

[See also same topic in AUJ 28-2 (Jun 2007), pp.76–77. —su]

Ahven — Unit test library

From: Tero Koskinen <tero.koskinen@iki.fi>
Date: Wed, 13 Aug 2008 18:53:58 +0300
Subject: ANN: Ahven 1.3
Newsgroups: comp.lang.ada

I would like to announce Ahven 1.3.

Ahven is a simple unit test library for Ada 95. It is loosely modelled after JUnit and some ideas are taken from AUnit. Ahven is distributed under permissive ISC license and should work with any Ada 95 compiler.

This is a bug fix release. For example, the changes include a few work arounds for Janus/Ada bugs and the removal of non-standard pragmas to let the source code compile cleanly on multiple different Ada compilers.

For more info, please read the detailed release notes:

http://home.gna.org/ahven/release_1_3.txt

or visit Ahven's homepage:

http://home.gna.org/ahven/

The source code is available as tar.gz and zip packages:

http://download.gna.org/ahven/ahven-1.3.tar.gz

http://download.gna.org/ahven/ahven-1.3.zip

Ps. Thanks to RR Software people for helping me with Janus/Ada issues and Debian Ada maintainers for creating Ahven's Debian packages.

[See also “AUnit Test Framework Experiences” in AUJ 24-2 (Jun 2003), p.81. —su]

Ada 83 support in Hibachi?

From: Tom Grosman <grosman@aonix.fr>
Date: Wed, 2 Jul 2008 10:42:30 —0400
Subject: RE : [hibachi-dev] Ada83 Support
To: hibachi-dev@eclipse.org

> it was mentioned that Future Features would include Ada 83 Support. Now my question is if this Support is already included or when it will be. If not I am also interested to include it.

There are no current plans to support specific Ada 83 toolchains. We have had requests to add support, so it is a “possible” future feature, but realistically, it is unlikely that it will be included in the project roadmap. We are of course more than willing to accept the functionality as a contribution, and provide coaching should someone wish to develop it.

Hibachi Project Lead

From: Tom Grosman <grosman@aonix.fr>
Date: Tue, 8 Jul 2008 15:15:36 —0400
Subject: [hibachi-dev] Hibachi Project Lead
To: hibachi-dev@eclipse.org

As some of you may alreay know, for the past year or so, I have been having a series of health related issues, negatively impacting my ability to devote significant time to the Hibachi project. While my situation has stabilized, the treatment regime I'm under is constraining in terms of time and energy.

Because of this, I am unable to be an effective project lead while at the same time performing the duties related to my job at Aonix. Therefore, I would like to nominate David Phillips to become project lead. I have talked to David and he is willing and enthusiastic about stepping up to the role. I'm sure David will be a great force in moving Hibachi forward in the future.

Assuming the Tools PMC and EMO agree to the choice of David as project lead, I will work to help him with the transition. I hope to be able to contribute to Hibachi in the future, though in a more limited way.

Tom Grosman

Hibachi Project Lead

From: Pat Rogers <rogers@adacore.com>
Date: Tue, 08 Jul 2008 15:09:01 —0500
Subject: Re: [hibachi-dev] Hibachi Project Lead
To: hibachi-dev@eclipse.org
Organization: AdaCore

On behalf of AdaCore, let me say that we are sorry to hear that this step is necessary but that we certainly both understand and hope that your health situation is resolved for the better as soon as possible. We all appreciate what you have done for the Ada community in starting the Hibachi effort and look forward to working with you in the future.

We at AdaCore, and I am sure the entire Hibachi team, welcome David to this role and will support him fully.

Pat and the AdaCore team

Basil — Mailing list server

From: Jordan Bettis <google1@hafdconsulting.com>
Date: Tue, 22 Jul 2008 01:00:12 −0700 (PDT)
Subject: ANN: Basil — Internet Message (email) and MIME library for Ada v 1.0
Newsgroups: comp.lang.ada

I'm working on a project to eventually produce a mailing list server in Ada. I've completed the first major 'deliverable' for the project, an email and MIME library in Ada 2005.

The project page is located at:

<http://hafdconsulting.com/libre/basil>

The library has the following capabilities:

- Serialization and unserialization of Messages, including those with MIME parts, and API methods to easily do common things with the message objects.
- An API for MIME children of message objects (and of other MIME entities), based on the Ada List container. Essentially the model is of a message object and a list of children, who can themselves have children. More abstractly, it is a n-tree. I provide a Cursor object for which you can choose one of two traversal strategies. One simply walks the children of one entity, and the other, 'recursive' strategy, walks the entire n-tree.
- Very flexible parsers for RFC 2822 Date, Address, and Message ID headers, and of RFC 2045 Content-Type and Content-Transfer- Encoding headers. Data structures to represent such objects (for the date it uses an Ada Time object), and means to serialize the objects into fully compliant message headers.
- Base64 and Quoted-Printable encoding and decoding.
- Encapsualization and unencapsualization of message headers containing non-ASCII values conforming to RFC 2047.

I've written a software specification for the library and a test suite. I'll be writing a user manual shortly as well, but I've also extensively commented the spec files to serve as a basis for the adabrowse reference manual.

My original software spec document didn't include support for RFC 2183 Content-Disposition headers, so they aren't in this version of the library, although they are rather important for managing MIME parts so I'll try to get that into the library soon. I'll probably hold off on writing the manual until that's done.

From: Jordan Bettis <google1@hafdconsulting.com>
Date: Tue, 22 Jul 2008 09:35:55 −0700 (PDT)
Subject: Re: ANN: Basil — Internet Message (email) and MIME library for Ada v 1.0

Newsgroups: comp.lang.ada

> Wow, this looks quite professional! Are you planning to merge your library into AdaCore's Ada Web Server (http://libre.adacore.com/aws), which already includes an SMTP client and server but apparently lacks the powerful mail handling capabilities of your library?

I hadn't considered merging with AWS. It seems mostly focused on RFC 2821 (SMTP) and you actually wouldn't need MIME in an RFC 2821 MTA, nor would you need SMTP in an MUA (or anything designed to run as a client to an MTA like a mailing list server). Where Basil could complement AWS is with its RFC 2822 structured header parsers, which an MTA would need.

Probably the best way to do it would be to make a glue package that can translate AWS header objects into Basil header objects, so you can use the structured header parsers, and then back again. I don't think that would be very difficult and I could probably get that in the next release.

From: Jordan Bettis <google1@hafdconsulting.com>
Date: Tue, 22 Jul 2008 12:43:57 −0700 (PDT)
Subject: Re: ANN: Basil — Internet Message (email) and MIME library for Ada v 1.0
Newsgroups: comp.lang.ada

> If you're planning to use AWS's SMTP client code, it looks like you're going to need this glue for your mailing list server anyway.

SMTP won't be needed for the mailing list. People want mailing list servers that are integrated with their existing MTA systems, they don't want stand-alone systems. Mailman, one should note, provides interfaces with all the popular Unix MTAs and it's quite a bit of their code base, they did not, last I looked, provide an integrated MTA. There *might* be some advantage to providing an integrated MTA particularly if AWS makes that very easy to do, but it would not be a very popular feature of the system at all.

I am considering using AWS for the web interface portion, but then again you have the same integration problem. People want their web applications to run behind the web server *they* choose (or at least Apache) they don't want a stand-alone system. There are some people around working on web interface systems for Ada. When I get to that phase I'll look at them, and also look into the possibility of making an AWS-based system integrate with Apache. But integration with existing web servers is a killer feature for any interface I choose.

> The reason I wasn't asking is not because I was interested in using your library; only because I think it is nice in general to consolidate small but related libraries into larger ones,

I take exception to your belief that Basil is a “small” library:

Totals grouped by language (dominant language first):

ada:	10823 (98.29%)
python:	146 (1.33%)
sh:	42 (0.38%)

(…)

> or at least host them in common places, so that users can find them more easily.

Well, I have a 'selfish asshole' motive for working on these projects and provide them as free software, that is to promote my business. And that is why they'll be hosted on my professional website. But I think inside a week if you put “MIME” and “Ada” into Google, that page will be the top hit, with this message and other forms of publicity. Mr. Hermann has already informed me that he has added it to his list of Ada resources, for instance. And of course, the Debian developers around here could certainly help make it easier for people to find and use it. :)

> Another benefit is to avoid inconsistencies and incompatibilities between libraries, so that a single program can use several libraries simultaneously. And, of course, to reduce duplication of work. Speaking of which, is there a good reason why Basil needs its own “header objects” different from AWS's “header objects”?

Compared to *everthing else* in RFC 2822 and MIME, unstructured headers are a very simple concept. They're just a key string, a colon, and a value string, terminated by a non-folded CRLF sequence. RFC 2822 specifies the CRLF but to make Basil more robust, I made it also handle lone CR or LFs.

So that was no great part of the library but it is a core part. To use AWS I would have had to integrate every layer with that system, making it a hard-dependency for building or using Basil, just so it could do the light lifting beneath all of Basil's heavy lifting.

Also, my header lists *are* integrated into the Doubly_Linked_List containers. My Headers.Lists.List object is a subtype of the DLL, so any operation you could perform on a DLL you can perform on my lists (except sorting isn't supported because of that interior generic). I also provide a custom equality function that considers things equal if their case insensitive keys are equal (per RFC 2822), this allows you do do something like use the Find operation to iterate over all headers with the same key.

I tried to make this library integrate very well with the Ada language, to leverage the power of the features that it has and provide an interface that mimics the behavior of relevant core interfaces that are already well-understood. I didn't see much use in integrating with vaguely related external third-party libraries, making the system more difficult to install and use.

But anyway, that's why I made the technical decisions I made. Of course writing software is the process of choosing between many different choices which all have their merits and downsides. Maybe you're right and I should have done this completely differently, there's not really any way to know for sure. I would like to thank you for your incredibly fast response to the bug I filed against gnatpp.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 22 Jul 2008 18:30:55 −0500
Subject: Re: ANN: Basil — Internet Message (email) and MIME library for Ada v 1.0
Newsgroups: comp.lang.ada

> Please post again to news:comp.lang.ada with reports of how well the mail server has been working. After some experience, it might be worthwhile for various Ada organizations to replace the solutions which they currently use with the new mailserver software. It might be trickier for ACM SIGAda which uses the ACM's central ListServ system, but I suspect that the the smaller scale operations could readily adopt the option from HAFD Consulting.

For what it's worth, R.R. Software and most of the AdaIC functions have been running on a primarily Ada mail server (and all Ada web server) for several years. I use an ancient public domain mail server for local mail delivery (POP3) and an ancient mailing list program to handle the mailing list, but all of the mail receiving, delivery, and spam filter functions are in Ada.

The question will be asked why it was never made available to the community, and the answer is that it wasn't designed to be that — it was intended to be a professional grade spam filter and as such the *code* wasn't documented much nor made very flexible. (In particular, there aren't any libraries that could be used in other applications, although I'm sure parts could be extracted.) There's plenty of user-level documentation, but that's it. (It's also a Claw application, and as such fairly tied to Windows for its user interface.)

Ada-related Products

AdaCore — GNAT-AJIS

From: AdaCore Press Center
Date: Tuesday June 17, 2008

Subject: Adacore Releases GNAT Ada-Java Interfacing Suite
RSS: http://www.adacore.com/2008/06/17/ada-java_interfacing_suite/

New tool suite helps developers create multi-language applications

VENICE, Italy, and NEW YORK, June 17, 2008 — Ada-Europe 2008 — AdaCore, provider of the highest quality Ada tools and support, today announced availability of the GNAT Ada-Java Interfacing Suite (GNAT-AJIS), which allows developers to build applications using both languages. With GNAT-AJIS, programmers can combine Java applications compiled to the Java Virtual Machine (JVM) and Ada code that has been compiled either natively or to the JVM. Application areas as diverse as financial services, communications, aerospace, defense, and academic research can all benefit from GNAT-AJIS.

GNAT-AJIS is aimed at both the Ada and Java development communities. For Ada programmers, GNAT-AJIS provides a mechanism to plug Ada components into systems (such as GUI frameworks) that are often written in Java. For Java programmers, GNAT-AJIS provides a means to take advantage of Ada's functionality or performance, for example in real-time control.

The initial release of GNAT-AJIS consists of two principal tools:

- A binding generator that takes an Ada package specification as input and produces Java classes as output, with native methods corresponding to the Ada subprograms
- The JGNAT compiler, which compiles Ada to the JVM.

Through the binding generator, Java applications can call native Ada code. The interfacing uses the Java Native Interface (JNI), but the binding generator produces the necessary "glue code" so that the programmer need not be concerned with the details. In JGNAT, since both Java and Ada are compiled to bytecodes, the interfacing is direct.

"With today's increasingly ambitious system requirements, multi-language systems are more and more common," said Robert Dewar, CEO and President of AdaCore. "The GNAT Ada-Java Interfacing Suite allows projects to realize the benefits of both Ada and Java through a cohesive framework. Ada is well suited to this level of co-operation, especially with the enhancements added in Ada 2005."

Pricing and Availability

The GNAT Ada-Java Interfacing Suite is available as an add-on to users of AdaCore's GNAT Pro development environment. Pricing for GNAT Pro subscriptions starts at $14,000. Please contact AdaCore (info@adacore.com) for the latest information on pricing and supported configurations.

About GNAT Pro

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

At the heart of GNAT Pro is a full-featured, multi-language development environment complete with libraries, bindings and a range of supplementary tools. All GNAT Pro technology is distributed with complete source code. GNAT Pro is based on the widely used GCC technology, is subjected to a rigorous quality assurance process, and is backed by rapid and expert support service.

Please contact AdaCore (info@adacore.com) for the latest information on pricing and supported configurations.

About AdaCore

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

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

[See also "JGNAT is coming back" in AUJ 29-2 (Jun 2008), p.83 and "JGNAT and MGNAT" in AUJ 27-3 (Sep 2007), pp.150–151. —su]

AdaCore — GNAT Component Collection

From: AdaCore Press Center
Date: Tuesday June 17, 2008
Subject: AdaCore Announces the Release of the GNAT Component Collection
RSS: http://www.adacore.com/2008/06/17/gnat_component_collection/

Suite of proven, reusable components now available to GNAT Pro customers

VENICE, Italy, and NEW YORK, June 17, 2008 — Ada-Europe 2008 — AdaCore, provider of the highest quality Ada tools and support services, today announced the release of the GNAT Component Collection, a suite of reusable software components and utilities. The GNAT Component Collection has been used by AdaCore in developing the GNAT Pro tool set, the GPS Integrated Development Environment, and the GNAT Tracker web-based customer interface, and is now available to GNAT Pro customers.

"The GNAT Component Collection contains a variety of software utilities that have proven to be extremely useful internally at AdaCore for product development," said Robert Dewar, President and CEO of AdaCore. "Since these are general-purpose components, we realized that they would also be of benefit to our customers. Thus we are making them available as part of the standard GNAT Pro subscription."

The GNAT Component Collection includes:

- Software that allows integration with scripting languages, such as python
- Database interfaces for APIs, such as postgresql, mysql, and sqlite
- Ada packages supplying a variety of services, such as
    - Module tracing
    - Efficient file IO
    - Efficient static string searching (Boyer-Moore algorithm)
    - E-mail and mailbox manipulation
    - Ravenscar tasking pattern examples
    - Various predefined storage pool utilities

"The GNAT Component Collection is, in effect, a software menu from which developers can select, à la carte, exactly those packages that are needed for their application," said Emmanuel Briot, lead project engineer for the GNAT Component Collection. "The collection consists of dozens of packages, interfaces and utilities that the GNAT Pro team has developed over the years. Offering this technology to our customers should allow them to realize the same productivity advantages that we have experienced at AdaCore."

Among the elements of the GNAT Component Collection is an extensive set of templates for Ravenscar Profile examples. These include:

- Simple_Cyclic_Task: a simple cyclic task that executes a given operation at a constant frequency
- Simple_Sporadic_Task: a sporadic task released by software invocations with a constant minimum inter-release time (in the worst case, its behavior is identical to a cyclic task)
- Sporadic_Server: a sporadic server that buffers a single type of request (carrying input parameters) and executes it enforcing a constant minimum inter-release time
- Sporadic_Server_With_Callback: a pattern that demonstrates how to model an asynchronous call with "out" parameters in Ravenscar.
- Multiple_Queue_Sporadic_Server: a Sporadic_Server variant that accepts multiple kinds of requests
- Sporadic_Server_With_Timeout: a Sporadic_Server variant that provides for the automated release of the server, if it is not released within a given amount of time by an explicit software invocation.

From: "Jamie Ayre" <ayre@adacore.com>
Date: Wed, July 30, 2008 10:56 am
Subject: [AdaCore] GNAT Components Collection beta now available
To: announce@adacore.com

AdaCore is pleased to announce the beta release of the GNAT Components Collection. It is compatible with all GNAT Pro releases starting with 6.0. Most of the components are target independent and can be used on any GNAT Pro supported platform although at this stage the beta is available on main native platforms such as x86-windows, x86-linux & sparc-solaris.

The GNAT Components Collection is a suite of reusable software components and utilities, and includes:

- Software that allows integration with scripting languages, such as python
- High level API to SQL databases (postgreSQL support provided, contact us for other systems)
- Ada packages supplying a variety of services, such as
  - Logging facility with support for colors, per module streams, syslog, etc.
  - Efficient file IO
  - Efficient static string searching (Boyer-Moore algorithm)
  - E-mail and mailbox handling
  - Ravenscar tasking pattern examples
  - Various predefined storage pool utilities

The GNAT Components Collection is available from the "Download GNAT Pro" section of GNAT Tracker. As always, for questions, or to inform us of issues that you encounter, please let us know through the GNAT Tracker report facility or by email at the usual report@adacore.com address.

AdaCore — GNAT Pro for RTX

From: AdaCore Press Center
Date: Tuesday July 15, 2008
Subject: AdaCore announces GNAT Pro for RTX
RSS: http://www.adacore.com/2008/07/15/rtx/

New product brings Ada to RTX for real-time development on Windows

NEW YORK and PARIS — July 15, 2008 — AdaCore, provider of the highest quality Ada tools and support, today announced the availability of GNAT Pro for RTX, an Ada Integrated Development Environment that enables programmers to produce real-time Ada applications on Microsoft Windows platforms. GNAT Pro for RTX supports two different modes: a Windows executable with memory protection, and a real-time subsystem that executes in kernel mode with hard, real-time behavior. This enhanced control and scalability helps simplify development of critical applications, including industrial automation, aerospace, and military systems.

RTX® software from Ardence, a Citrix Company, enables Windows systems to deliver hard, real-time performance. As a true extension, it does not interfere with or modify the Windows infrastructure. Developers can create user interfaces and applications that take advantage of all the functionality offered by Windows. A component that requires real-time control can first be developed and debugged as a Windows application and then recompiled as a real-time subsystem with no code changes.

"Ada and AdaCore have a well-deserved reputation for excellence in the real-time domain," said Jeffrey D. Hibbard, who heads the Embedded Group for Ardence, a Citrix Company. "We are pleased that GNAT Pro is available on RTX, allowing customers to reap the productivity benefits that come from Ada's modern features and AdaCore's front-line support."

"This new RTX port stems from customer requests for an Ada development environment that takes advantage of all the functionality of RTX without requiring a rewrite of their code," said José Ruiz, AdaCore senior software engineer. "GNAT Pro for RTX allows our customers to take Windows applications, recompile them unchanged, and get predictable, hard real-time behavior without interference from the underlying Windows Operating System."

"Two types of customers will benefit from this product," said Robert Dewar, President and CEO of AdaCore. "The first will use Windows plus RTX as a testing and development platform to verify real-time properties of their applications before migrating to another real-time target. The second will use RTX to execute real-time applications in kernel mode without having to pass through Windows to access the device's memory, etc. Either way the GNAT Pro solution will help our customers reduce the development costs for their real-time projects."

Pricing and Availability

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

Windows is a registered trademark of Microsoft Corporation in the United States and other countries. RTX is a trademark of Citrix Systems, Inc. and/or one or more of its subsidiaries. All other trademarks and registered trademarks are property of their respective owners.

[See also "AdaCore — GNAT Pro for VxWorks 653" in AUJ 29-2 (Jun 2008), p.86. —su]

AdaCore — GNAT Pro 6.1.2

From: "Jamie Ayre" <ayre@adacore.com>
Date: Thu, July 17, 2008 8:55 pm
Subject: [AdaCore] Announcing the availability of GNAT Pro 6.1.2
To: announce@adacore.com

AdaCore is pleased to announce the immediate availability of the GNAT Pro 6.1.2 release for the following native platforms:

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

and the following cross platforms:

erc32-elf-linux
erc32-elf-solaris
leon-elf-linux
leon-elf-solaris
ppc-elf-solaris
ppc-elf-windows

Other platforms will be made available in the coming weeks.

GNAT Pro 6.1.2 provides fixes for issues reported in the 6.1.1 release and documented in the known-problems-611 file (available through GNAT Tracker).

We also are in the process of transitioning the GNAT Pro technology to a new compiler back-end based on GCC 4.3 which we expect to bring performance improvements to user applications. Our goal is to have several of our supported configurations on this back-end for the next major GNAT Pro release scheduled early 2009.

In the coming weeks, we will provide beta versions for several platforms and would appreciate feedback on this new technology from interested GNAT Pro users.

GNAT Pro 6.1.2 is available from the "Download GNAT Pro" section of GNAT Tracker. As always, for questions, or to inform us of issues that you encounter, please let us know through the GNAT Tracker report facility or by email at the usual report@adacore.com address.

[See also "AdaCore — GNAT Pro 6.1.1" in AUJ 29-1 (Mar 2008), pp.12–13. —su]

Praxis HIS — SPARK Toolset 7.6

From: SPARKAda.com
Date: July 2008
Subject: SPARK Release 7.6
URL: http://www.praxis-his.com/sparkada/release7p6.asp

Praxis High Integrity Systems is pleased to announce the immediate availability of Release 7.6 of the SPARK language and the SPARK toolset.

Full details of all language and tool changes can be found in the release notes for release 7.6.

Supported, professional customers will receive upgrade packages immediately.

Buyers of the "SPARK Book" by John Barnes can now download upgrade packages to bring their toolset and documentation up to release 7.6.

Release 7.6 includes many significant improvements, including:

- Corrections to the VC Generator for narrowing subtype conversions involving enumerated types.
- Record types that have a single field which is a predefined scalar type are now allowed to be Atomic in RavenSPARK mode.
- Improvement to the default-invariant generator where a variable controlling a dynamic "for" loop is know to be mode "in".
- Simpler FDL modelling of the 'Pos and 'Val attributes of type Character.
- Better modelling of T'Valid for a subtype T in proof rules.
- New "Output_Directory" option for the Examiner.
- New "order" option for SPARKFormat that allows for alphabetic or declaration-ordering of reformatted annotations.
- Improved VC Generation for local variables in subprograms that indirectly import an external own variable.
- Improved performance in the Simplifier where user-defined proof rules give rise to conditions that are fully instantiated.
- New options on SPARKMake to suppress generation of the index and/or meta-files and to process sources that don't have a main subprogram.

Please email us for more information at sparkinfo@praxis-his.com

[See also "Praxis HIS — SPARK Toolset 7.5" in AUJ 28-2 (Jun 2007), p.84. —su]

Ada and GNU/Linux

Debian 5.0 "Lenny"

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Tue, 19 Aug 2008 07:53:54 +0200
Subject: Ada in Debian 5.0 "Lenny"
Newsgroups: comp.lang.ada

Debian 5.0 "Lenny" is due for release in September and has been frozen since the end of July[1] but, due to lack of time on my part, some Ada packages didn't make the deadline and are now in unstable. I requested[2] from the Debian release team an exception to the freeze so these packages can migrate to Lenny and I am awaiting a response.

[1] http://lists.debian.org/debian-devel-announce/2008/07/msg00007.html

[2] http://lists.debian.org/debian-release/2008/08/msg00702.html

The new default Ada compiler is gnat-4.3. In addition to the many bug fixes and enhancements in GCC 4.3.1[3], the version shipped with Debian contains 19 bug fixes backported from the main line of development (GCC 4.4). This is the result of Samuel Tardieu's outstanding work. Debian also pioneered support for mips, mipsel and ppc64 architectures thanks to Xavier Grave, Aurelien Jarno and Andreas Jochens. This support will be included in GCC 4.4. Finally, gnat-4.3 ships with two versions of the Ada run-time library: one using zero-cost exception handling (aka ZCX, both static and shared libraries), and one using the setjump/longjump (SJLJ) mechanism (static library only).

According to the popularity contest[4], gnat-4.1 users are already migrating en masse to gnat-4.3. I am very happy about this and I take it as a personal encouragement to continue work on Debian.

[3] http://gcc.gnu.org/bugzilla/…

[4] http://people.debian.org/~igloo/…

Packages recompiled with gnat-4.3 but no other changes:

- ada-reference-manual — The standard describing the Ada 95 language
- adabrowse 4.0.2 — HTML generator for Ada 95 library unit specifications
- adacgi 1.6 — Ada CGI interface
- libaunit 1.03 — AUnit, a unit testing framework for Ada

Packages updated:

- adacontrol — An Ada rules controller: 1.6r8 -> 1.9r4
- adasockets — bindings for socket services in Ada: 1.8.4.7 -> 1.8.6
- asis — Ada Semantic Interface Specification: 2005 -> 2007
- gnade! — GNU Ada Database Environment: 1.6.1 -> 1.6.2
- gnat-gps! — The GNAT Programming System: 4.0.1 -> 4.3 (prerelease)
- libaws — Ada Web Server: 2.2 -> 2.5 (prerelease)
- libflorist! — POSIX.5 interface to operating system services: 2006 -> 2008
- libgtkada2! — Development files for libgtkada2: 2.8.1 -> 2.12.0 (prerelease)
- libtemplates-parser — Ada library to parse files and replace variables with their values: 10.0+20060522 -> 11.1
- libtexttools — Ada and C++ library for writing console applications: 2.0.3 -> 2.0.5
- libxmlada — XML/Ada, a full XML suite for Ada programmers: 2.2 -> 3.0

The packages marked ! above are awaiting approval to migrate into Lenny before the release; they are now in unstable.

New Ada packages:

- ahven — Unit test library for Ada: 1.2
- ghdl — VHDL compiler/simulator using GCC technology: 0.26+svn98
- libalog — Logging framework for Ada: 0.1
- plplot-ada — Ada support for PLplot, a plotting library: 5.9.0
- topal — Links Pine and GnuPG together: 68

Many thanks to Reto Buerki and Andrew Ross for these new packages. (ghdl and topal are not really new; they were present in Debian 3.1 "Sarge" but not in 4.0 "Etch", and are re-added in 5.0 "Lenny").

The sad news is that Lenny will not support distributed programming in Ada. Xavier Grave has already explained why[5].

Packages removed:

gnat-glade — GNAT Library for Ada Distributed Execution

gnat-glade-doc — GNAT Library for Ada Distributed Execution

[5] Xavier Grave, "Ada in Debian: dropping support for the Distributed Systems Annex", comp.lang.ada, 2007-07-28.

[See also "Debian — Transition to GCC 4.3" in AUJ 29-1 (Mar 2008), p.16 and "Debian — Package gnat-glade removed" in this issue. —su]

Debian — Package gnat-glade removed

From: Xavier Grave <xavier.grave@ipno.in2p3.fr>
Date: Mon, 28 Jul 2008 08:38:56 +0200
Subject: Ada in Debian: dropping support for the Distributed Systems Annex
Newsgroups: comp.lang.ada

The package gnat-glade in Debian has been providing support for Annex E (the Distributed Systems Annex) since 1999. With this package, it is possible to write programs that are distributed over many computers.

I have found a critical bug that makes the version currently in testing (gnat-glade 2007) unusable for real-world applications. The problem is described in detail in [1]. Now that AdaCore has abandoned GLADE in favour of PolyORB, there will be no solution to this problem.

Therefore, starting in April 2008, I packaged PolyORB 2.3 for Debian. During testing, I discovered other problems [2] that make PolyORB unusable too. Consequently I decided not to upload the package to Debian, but to wait for the next release.

After AdaCore released PolyORB 2.4 (as part of GNAT GPL 2008 Edition) I resumed work but the problems are still there, whether using gnat-4.3 or GNAT GPL 2008 as the compiler [3].

As a consequence, I am sorry to announce that Debian 5.0 "Lenny", scheduled for release in September 2008, will not support the Distributed Systems Annex.

I will continue working on PolyORB in Debian as my time allows. If you think you can help, please do not hesitate to get in touch with me. The packaging scripts are on Ada-France's public monotone server [4,5]. Maybe one day I will be able to integrate a rock-solid PolyORB in some future version of Debian.

[1] http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=492100

[2] see thread started by :

http://lists.adacore.com/pipermail/polyorb-users/2008-May/000950.html

[3] http://lists.adacore.com/pipermail/polyorb-users/2008-June/000984.html

[4] http://www.ada-france.org:8081/branch/changes/org.debian.polyorb

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

From: Ludovic Brenta <ludovic@ludovic-brenta.org>
Date: Wed, 30 Jul 2008 10:15:16 −0700 (PDT)
Subject: Re: Ada in Debian: dropping support for the Distributed Systems Annex
Newsgroups: comp.lang.ada

> It means, It doesn't even work it to give a try compile it under Debian?

gnat-glade itself compiles (there is a binary package in Debian testing) but if you try to compile your own distributed programs with it, you get compiler errors due to version mismatches between compiler (gnat-4.3), run-time library (libgna-4.3) and libgarlic-dev. These mismatches are detected by means of the .ali files.

The state of PolyORB is that both PolyORB itself and the distributed program compile fine, but then the distributed program gives errors at run time.

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 31 Jul 2008 09:09:03 +0200
Organization: Adalog
Subject: Re: Ada in Debian: dropping support for the Distributed Systems Annex
Newsgroups: comp.lang.ada

Point of information: Glade has reappeared in GnatPro 6.1.2

I guess this means that there /are/ paying customers who use Annex E…

References to Publications

Functional programming in Ada

From: Harald Korneliussen <vintermann@gmail.com>
Subject: Functional programming in Ada
Date: Thu, 3 Jul 2008 12:36:24 −0700 (PDT)
Newsgroups: comp.lang.ada

FP guru Chris Okasaki wrote an interesting post on doing functional programming in Ada. Perhaps mostly a curiosity, but interesting that it can be done.

http://okasaki.blogspot.com/2008/07/functional-programming-inada.html

Embedded Control Europe — "Code coverage"

From: AdaCore Press Center
Date: Tuesday June 24, 2008
Subject: Code coverage: free software and virtualization to the rescue
RSS: http://www.adacore.com/2008/06/24/code-coverage-free-software-and-virtualization-to-the-rescue/

Embedded Control Europe

[See http://www.embedded-control-europe.com/know-how?kid=206 and http://www.embedded-control-europe.com/c_ece_knowhow/206/basapr08p32.pdf —su]

SAE International — "Conflict detection software runs on Ada"

From: AdaCore Press Center
Date: Wednesday July 23, 2008
Subject: Conflict detection software runs on Ada
RSS: http://www.adacore.com/2008/07/23/conflict-detection-software-runs-on-ada/

SAE International

COTS Journal — "Real-Time Concurrent Issues Drive Ada versus Java Choice"

From: AdaCore Developer Center
Date: Tuesday August 26, 2008
Subject: Real-Time Concurrent Issues Drive Ada versus Java Choice
RSS: http://www.adacore.com/2008/08/26/real-time-concurrent-issues-drive-ada-versus-java-choice/

Ben Brosgol has recently published a paper in COTS Journal that examines the real-time advantages of Ada and Java:

"On the surface, Ada and Java offer similar features to support real-time embedded military applications. But under the hood, they differ significantly in their underlying philosophy."

To read the full article, please click here Real-Time Concurrent Issues Drive Ada versus Java Choice or visit

http://www.cotsjournalonline.com/home/article.php?id=100827

AdaCore — Embedded Systems Conference 2008

From: AdaCore Press Center
Date: Wednesday August 13, 2008
Subject: Embedded Systems Conference (ESC) 2008
RSS: http://www.adacore.com/2008/08/13/embedded-systems-conference-esc-2008/

AdaCore will be exhibiting at this event.

AdaCore — SIGAda 2008

From: AdaCore Press Center
Date: Wednesday August 13, 2008
Subject: SIGAda 2008
RSS: http://www.adacore.com/2008/08/13/sigada-2008/

AdaCore will be exhibiting at this event.

Keynote: AdaCore senior engineer, Ben Brogsol, will give the keynote address entitled, 'From Strawman to Ada 2005: a Socio-Technical Retrospective'. (Sunday October 26)

Tutorial: Ben Brogsol will present a tutorial on 'Languages for Safety-Critical Software: Issues and Assessment'. (Tuesday October 28)

AdaCore — SCADE User Group Conference

From: AdaCore Press Center
Date: Wednesday July 23, 2008
Subject: SCADE User Group Conference
RSS: http://www.adacore.com/2008/07/23/scade-user-group-conference/

AdaCore is an invited partner at this event and will be exhibiting.

Rapita — Embedded Systems Show (ESS08)

From: Rapita News
Date: 1 September, 2008
Subject: Visit Rapita Systems at the Embedded Systems Show (ESS08), NEC, Birmingham, UK 1st-2nd October 2008
RSS: http://www.rapitasystems.com/node/342

Rapita Systems will be attending the Embedded Systems Show (ESS) at the National Exhibition Centre, Birmingham, UK. We are exhibiting with SDC Systems, our UK distributor on booth 540. Come and visit us there.

Further details of the conference can be found on the Embedded Systems Show website. http://www.embedded.co.uk/index.php

Ada Inside

Who's Using Ada in industry?

From: Michael Feldman <mfeldman@gwu.edu>
Date: Tue, 10 Jun 2008 12:52:48 −0700
Subject: "Who's Using Ada" — June 2008 edition
Newsgroups: comp.lang.ada

"Who's Using Ada: Real-World Projects Powered by the Ada Programming Language"

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

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

I've decided to leave the previous editions online for comparison purposes. The basic link above will always point to the most recent; I've identified the older ones by adding a year and month to the file name. For example, May 2008 is http://www.seas.gwu.edu/~mfeldman/ada-project-summary-0805.html

I'm still checking out some tips and possible links; they'll appear in the July edition, which will be delayed till mid-July because I'm traveling.

Have a good June; see you again in July.

[See also same topic in AUJ 29-2 (Jun 2008), p.91. —su]

Ipesoft Selects GNAT Pro to Develop Real-Time Management Systems

From: AdaCore Press Center
Date:Tuesday June 17, 2008
Subject: Ipesoft Selects GNAT Pro to Develop Real-Time Management Systems
RSS: http://www.adacore.com/2008/06/17/ipesoft/

Ipesoft Selects AdaCore and GNAT Pro to Develop Real-Time Management and Production Systems

VENICE, Italy, and NEW YORK, June 17, 2008 — Ada-Europe 2008 — AdaCore, provider of the highest quality Ada tools and support services, today announced that Ipesoft, s.r.o, a leading developer of real-time applications and long-standing AdaCore customer, has extended the use of GNAT Pro on its high-reliability servers. Ipesoft is now using GNAT Pro to develop D2000® Enterprise Production Systems, a unique family of solutions that provides real-time management and production systems for manufacturing and energy facilities.

Based in the Slovak Republic, Ipesoft is focused on two areas — developing technology for real-time applications, and acting as a systems integrator to create and incorporate complete mission-critical solutions for its customers, based on its D2000 technology. D2000 enables the creation of integrated solutions for the whole enterprise — from process automation, comprehensive monitoring and management of production processes, to integration with other information systems within the enterprise. The core of this technology is a real-time application server with robust platform-independent RAD (Rapid Application Development) tools. The server is scalable, distributed, and supports redundancy. The server code is written in Ada.

In the industry targeted by Ipesoft, reliability and long-term maintenance are key. The Ada programming language satisfies these core needs by offering real-time features and support for distributed applications, strong type checking that helps detect errors and avoid vulnerabilities, and language stability that has maintained backwards compatibility at every evolution of the language (Ada 2005, Ada 95, and Ada 83). These attributes are vital given the size of the D2000, which has grown to encompass over 1.7 million lines of code since its inception in 1993. GNAT Pro and AdaCore were chosen due to the combination of multi-platform availability (GNAT Pro is available on more platforms than any other Ada development environment) and the strong support that AdaCore provides.

"Ipesoft's adoption of Ada for its core product development demonstrates the advantages that the language provides for long-lived, mission-critical applications," said Cyrille Comar, Managing Director, AdaCore Europe. "Using GNAT Pro has enabled Ipesoft to create real-time production and control systems that underpin the company's growth and deliver high integrity solutions to its growing customer base."

"D2000 is the cornerstone of all Ipesoft's current and future activities," said Miroslav Kunsch, CEO, Ipesoft. "We needed a long-term partner and a scalable software development environment. Selecting AdaCore has been key to our success; choosing Ada was the right decision at the right time and we are glad to reinforce our relationship with this new contract."

Ipesoft began work with AdaCore in 2002, initially using GNAT Pro for Windows. It is now using GNAT Pro for Itanium OpenVMS, GNAT Pro for Alpha OpenVMS and GNAT Pro for x86 Windows. The GNAT Pro development environment combines market-leading technology with an expert support system to provide a natural solution where efficient and reliable code is critical.

About Ipesoft

Ipesoft is a pioneer and leader in the production systems segment of the Slovak market. Ipesoft products provide solutions for all aspects of manufacturing enterprises, from processing automation to complex monitoring and controlling of production processes, which can be integrated in Enterprise Resource Planning (ERP) company information systems. Solutions based on D2000 products are powerful tools, helping management at all levels. Managers receive quality, reliable and up-to-date information for operational planning. The richness and quality of information aids the decision-making process, thereby bringing truly effective manufacturing enterprise control. Ipesoft solutions are helping companies to increase productivity through production process mapping and indicating areas needing improvement.

Ipesoft integrates data from multi-location plants into a comprehensive database allowing visualization of performance of all the facilities as a whole. The system communicates with other information systems within the enterprise like ERP, Supply Chain Management (SCM), etc. Ipesoft solutions are designed as open, flexible and ready to grow according to our customers' requirements. Keywords here are collaborating manufacturing, complete knowledge-based solution exceeding expectations, and standard Manufacturing Execution System (MES) capabilities.

Ipesoft's staff provides a comprehensive set of services, ranging from analysis, application concept, design and implementation, to post-implementation support.

Robins Air Force Base Selects DDC-I for AC-130U Software Support

From: DDC-I Press Releases Archive
Date: August 26, 2008
Subject: Robins Air Force Base Selects DDC-I for AC-130U Software Support
URL: http://www.ddci.com/display_news_item-filename-news_Robins_Air_Force_Base_Selects_DDC-I_for_AC-130U_Software_Support_release.htm

Replaces Legacy compilers with DDC-I's Eclipse-based OpenArbor IDE and SCORE-Ada compilers

Phoenix, AZ. August 26, 2008. DDC-I, a leading supplier of development tools for safety-critical applications, today announced that is has been selected by the US Air Force Warner Robins Air Logistics Center to assist with the USAF's organic refresh and support effort for the C130's avionics software. The USAF will use DDC-I's Open Arbor products to replace legacy MIPS and 1750A Ada compilers. As part of the compiler evaluation, the USAF has migrated one of the avionics applications to DDC-I's OpenArbor development environment.

"The Air Force has traditionally outsourced its support to contractors," said Bob Morris, president and CEO of DDC-I. "Now they are creating their own 'organic' software teams to save costs and reduce turn-around times. DDC-I is proud to provide foundational products, training, and services that will better enable the USAF to bring this work internally and ramp up its program teams in short order."

Starting with a clean slate enables the USAF software teams to look beyond the very basic tools that the original contractor used to developed and maintain the C130's avionics software. Now, instead of staying with rudimentary tools like command-line debuggers, Solaris/VAX servers, and weak profilers, the USAF teams can employ leading edge IDEs and other tools. These best-in-class tools give the USAF teams a technical advantage over the contractor's old-line approach, thereby enhancing productivity and robustness.

OpenArbor is a mixed-language, object-oriented IDE for developing and deploying real-time, safety-critical applications. The core environment combines optimizing compilers and libraries for C and Embedded C++ with the SCORE mixed-language debugger. The SCORE debugger features an intuitive multi-window GUI, project management support, and automated build/make utilities. SCORE's symbolic debugger recognizes C/EC++, Ada and Fortran syntax and expressions, and can view objects, expressions, call chains, execution traces, interspersed machine code, machine registers, and program stacks.

OpenArbor provides a separate Eclipse plug-in for Ada development. Known as SCORE-Ada, this plug-in 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.

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, JOVIAL and Fortran 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 or visit www.ddci.com.

Indirect Information on Ada Usage

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

Job Description: Belgium

Study/improve the techniques and tools available in the framework of valgrind and apply to a big Ada based application.

(…) making extensive use of Ada, including features such as Ada tasking (i.e. multi-threading) and dynamic memory.

Valgrind is a framework in which various checking tools have been developped for dynamic analysis. Among others:

- memcheck: checker for memory leak, dangling pointer, uninitialized data;
- cachegrind/callgrind: tool to find where CPU is used;
- massif: tool to report about heap usage;
- omega: instant leak detector;
- helgrind: tool searching for data race condition access;
- drd: similar to helgrind.

(…) The tools and how they work could be looked at and evaluated gradually:

- First on some very small executable tests: the ACATS tests (these are the standardized tests used for Ada compiler conformity checking). Running the ACATS tests under memcheck has already been done. These tests can be used for a first evaluation of other valgrind tools. (…)

Job Description: France

Activities:

- Design and implement middleware for distributed systems
- Specify middleware modules and services according to the specific needs of the distributed application
- Contribute to the existent middleware created by the group
- Contribute to research projects, particularly in the domains of real-time embedded systems, massively parallel systems and ad hoc networks.
- Specify and model a middleware and related services for deployment and verification of the distributed system.
- Port and adapt software to new machines and new systems.
- Assist and support users of the developed applications
- Participate in writing of documentation and project proposals in English and French.

Skills:

- Knowledge and experience in systems and networks: Unix, TCP/UDP, SQL
- Knowledge and experience in programming: C, Java, (Ada appreciated)
- Knowledge and experience in

distribution technologies: CORBA, RMI,
- Experience in software specification and modeling: UML, ADL,
- Experince in algorithms for distributed systems: global state, distributed memory
- Experience in verification: petri nets, synchronous languages
- Degree equivalente to a Master in informatics

Context:

In a department responsible for teaching and research in informatics and networks of a school of engineers leader in the field of telecommunications

In a group specialised in software development (10 Professors and 10 PhD students) and strongly connected to the competitiveness clusters.

[Translated from French —su]

Job Description: UK

I have a contract opportunity for a client based in South East, UK who is looking for an Ada contractor, initially for 3 months. The hourly rate is between £40 and £42 per hour.

Key skills for the role are:

- Ada 95
- Object Oriented Experience
- SC Security Clearance or eligible to obtain British Security Clearance
- Team Player
- Good communication skills

Desirable skills:

- UML
- Ada 83
- C++

Job Description: USA

The Senior Software Engineer will be responsible for development of embedded command, control, and communications software for the digitized battlefield. (…) It performs data messaging, provides wireless communication protocols, and interfaces with military communication equipment and aviation mission systems. Providing advanced applications, the software manages current battlefield situational awareness data on friendly and enemy forces for pilot display, and manages current mission orders and digital map overlays.

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

Education

BS Computer Science or other field

Required Skills

- Minimum of 7 years development experience with Ada 95
- Minimum of 9 years experience in software development.
- Must be able to work independently with minimal supervision.
- Experience with multi-threaded development.
- Good foundation and experience in Object Oriented Design.
- Experience developing software for LynxOS, Linux, VxWorks or Unix based targets.

Desired Skills

- Embedded systems development experience
- Experience with C/C++
- Experience with CORBA
- Experience with ClearCase

Ada in Context

Unix daemon in Ada

From: Jacob Sparre Andersen <jspa@nykredit.dk>
Date: Tue, 29 Jul 2008 02:05:52 −0700 (PDT)
Subject: Re: Ada daemon
Newsgroups: comp.lang.ada

> My good friend Dwight and I are wondering about how to create a daemon in Ada. So far we've only been able to locate this example:
>
> http://www.pegasoft.ca/resources/boblap/16.html#16.26
>
> which honestly looks both hideous and unpleasant, to us.

Systems programming isn't always beautiful. The trick is to notice that almost all the code you see is actually a reusable library. Also, several of the subprograms in the library do actually exist in the POSIX packages, so there's no need to manually bind to the C versions of the subprograms.

In practice the important part of daemonizing a process is to disconnect Standard_Input, Standard_Output and Standard_Error. You can do this with:

```
POSIX.IO.Close (File =>
  POSIX.IO.Standard_Input);
POSIX.IO.Close (File =>
  POSIX.IO.Standard_Output);
POSIX.IO.Close (File =>
  POSIX.IO.Standard_Error);
```

> Yes, it might be because we're just a bunch of beginners, but still we were hoping for a "prettier", more Ada like approach. This one seems to rely on a lot of pragma import C stuff.

Using the POSIX standard packages definitely makes it prettier.

> So, is this the only way to create a simple daemon, or can you point us in other directions?

Here's an (untested) implementation of Daemonize using "pure" Ada:

```
with
  POSIX.IO,
  POSIX.Process_Environment,
  POSIX.Process_Identification,
  POSIX.Unsafe_Process_Primitives;

procedure Daemonize is
begin

  POSIX.Process_Environment.
      Change_Working_Directory ("/");
  for File in
    POSIX.IO.File_Descriptor'Range
  loop
      if POSIX.IO.Is_Open (File) then
        POSIX.IO.Close (File);
      end if;
  end loop;

  case POSIX.Unsafe_Process_
      Primitives.Fork is
    when -1 =>
      POSIX.Process_Primitives.
           Exit_Process
      (POSIX.Process_Primitives.
       Failed_Creation_Exit);
    when 0 =>
    declare
       Session_Leader :
         POSIX.Process_Identification.
         Process_Group_ID;
     begin
        POSIX.Process_Identification.
         Create_Session
               (Session_Leader);
    end;
    when others =>
       POSIX.Process_Primitives.
         Exit_Process;
  end case;
end Daemonize;
```

On Debian you need to install the package "libflorist-dev", to have access to the POSIX packages.

Ichbiah and OOP in Ada 95

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Wed, 04 Jun 2008 16:14:21 +0200
Subject: Re: The A-Z of Programming Languages: Ada, interview with S. Tucker Taft
Newsgroups: comp.lang.ada

Has Ichbiah been right to be sceptical of Ada 95's OO? It does seem to have, uhm, rich structure, seen from a learners point of view.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Thu, 05 Jun 2008 15:58:36 +0200
Subject: Re: The A-Z of Programming Languages: Ada, interview with S. Tucker Taft
Newsgroups: comp.lang.ada

> This prompts the question: how would Ichbiah have implemented OOP in Ada?
>
> Perhaps barely, if at all.

This seems surprising because classwide programming and dynamic dispatch seem to have been among the requirements of Ada 9X. (I got this idea from browsing the archives.)

In an excerpt from a letter sent by Ichbiah to the Ada 9X group, a part of which is quoted in Meyer's OOSC2, Ichbiah appears to be addressing the increased complexity of the language, which is caused by the then new features and their combinations. He computes it to be approaching ~60_000 combinations. (I'll look up the details.) Might Ichbiah have thought that by some 80/20 rule, you can overdo things if you create Ada 9X the way it was around 1992?

IIUC what Taft says in the interview, Ichbiah didn't like the _way_ OOP was to be implemented. OTOH he had been working on a Simula compiler at INRIA. So maybe OOP alone was not the elephant.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 11 Jun 2008 20:57:41 −0500
Subject: Re: The A-Z of Programming Languages: Ada, interview with S. Tucker Taft
Newsgroups: comp.lang.ada

> I wasn't there, but rumor had it that JDI wanted to use the term "class" (rather than "tagged type") for consistency with other OO languages. I don't know what his other disagreements were.

I *was* there, and that was a major contention between two groups. (I was on Ichbiah's side on this argument, but I didn't quit when I lost. ;-). I don't recall any technical objections that he had. The whole discussion degenerated into something rather theoretical. I don't want to go into detail, because I don't want to speak ill of the dead (or of the living, for that matter).

I personally thought that using derived types (which no one understood in Ada 83) to implement classes was a mistake. Tucker was adamant on this point. I'm still not sure if he was right, but I'm used to it now. (Adding overriding indicators surely helps a lot, by indicating the programmers intent.)

From: Christoph Grein <christoph.grein@eurocopter.com>
Date: Thu, 12 Jun 2008 00:25:25 −0700 (PDT)
Subject: Re: The A-Z of Programming Languages: Ada, interview with S. Tucker Taft
Newsgroups: comp.lang.ada

(…) I used type derivation a lot in Ada 83 with much benefit for handling physical types adding new operations on the derived type.

And there is the trick attributed to John Goodenough to add user-defined equality to any type.

(…) I think that using the Ada 83 type derivation facility for this is very natural. Why invent a completely new technique?

From: Jean-Pierre Rosen <rosen@adalog.fr>
Subject: Re: The A-Z of Programming Languages: Ada, interview with S. Tucker Taft
Date: Thu, 05 Jun 2008 16:13:59 +0200
Newsgroups: comp.lang.ada
Organization: Adalog

Actually, Ichbiah was well aware of the benefits of OOP, and actually that's why he insisted for having derived types in Ada 83, against the opinion of the rest of his team.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Thu, 05 Jun 2008 21:40:55 +0200
Subject: Re: The A-Z of Programming Languages: Ada, interview with S. Tucker Taft
Newsgroups: comp.lang.ada, comp.lang.eiffel

As promised, here is what Meyer quotes from Ichbiah's resignation letter:

"A massive increase in complexity will result from 9X adding one or more additional possibilities where Ada now offers two. For example, 9X adds: […] access parameters, to IN, OUT, and IN OUT; tagged types, to normal types; dispatched subprogram calls, to normal subprogram calls; use type clause, to use package clauses; … With 9X, the number of interactions to consider is close to 60,000 since we have 3 or more possibilities in each case (that is, 3^10)." (OOSC2, §33.7, p.1095)

The comment "adds: tagged types, to normal types" is particularly interesting, I think, because it touches on a consequence of this distinction: sloppy versus exact base type systems:

Eiffel tries to have basically one kind of type definition, namely the class—even though "expanded class" "adds one or more additional possibilities", if I may aim Ichbiah's comment at Eiffel. Compiler magic is/was used for types such as INTEGER or REAL. Some operations of INTEGER are "require"-predicates used for testing whether or not an integer value fits a subsets of INTEGER, e.g. 8-bit integers.

Ada, as mentioned by Ichbiah, has "normal" types for defining integers, reals, etc., and tagged types for defining polymorphic types. You want integers between 0 and 10_000 only? Define a corresponding normal type, or do "normal" derivation from another integer type adding the needed constraint. (Part of the language since Ada 83 as pointed out by J.-P. Rosen above.)

What I find so interesting is that these ways to define basic types might show that there are undeniable reasons to require two type definition mechanisms. (I guess this is not news to people who worked on either Ada 9X or Eiffel, but it appears to be news to a new audience tackling the base type system. And the presence of one or the other always affects programs.)

Do the Eiffel base types work well? Do they match the "normal" integer types of Ada in practice? Using cut&paste polymorphism and renaming one can change INTEGER to a different INTEGER with more specific require predicates, predicates even more powerful than Ada's range constraints(*). Still, people coming to Eiffel have more than once asked for more programmer control of basic Eiffel types such as INTEGER and FLOAT. Messing with base types, renaming and cluster management do not look like the best solution.

So maybe there is good reason to have both normal types, and tagged types, even if this complicates the language?

(*) Some hard work has been done with the goal of enhancing Ada's type constraints in the sense of DbC. It has been published as AIs and elsewhere.

From: Georg Bauhaus <bauhaus@futureapps.de>
Date: Fri, 06 Jun 2008 19:57:26 +0200
Subject: Re: The A-Z of Programming Languages: Ada, interview with S. Tucker Taft
Newsgroups: comp.lang.ada

> In which sense "sloppy/exact"?

Sloppy/exact in the sense of mapping the set of problem or solution values to a set of type values 1:1, bijectively.

When all you have is int, INTEGER, Int, etc. but your set of whole numbers has bounds strictly inside (min machine-int, max machine-int), say, then one type system allows you to exactly give the lowest and highest whole number that your (sub)type is to have. Another type system, namely that of Qi, even permits computing a set of values using a Turing complete type declaration language. So "only odd natural numbers" will be a perfectly normal Qi type. A type definition that basically must use "int", but the set of values is between 0 and 1_000_000, is sloppy in that it does not express the set, neither to the reader nor to the compiler.

You might recall the report that an embedded systems teacher from a US university has summarized here. A base type system that is a more exact representation of the solution sets does help in programming.

> Tagged types aren't polymorphic. Only their classes (closures of) are.

Yes, and you define tagged types in order to get classwide polymorphic types. There are no class-wide types rooted at some non-tagged type. There is a universal type. This, I think, reflects the tagged vs normal distinction Ichbiah is listing. (…)

From: Robert A Duff <duff@adacore.com>
Date: Tue, 15 Jul 2008 20:15:05 −0400
Subject: Re: Ada OOP alternatives?
Newsgroups: comp.lang.ada

> might anybody know what the different designs were for OO in Ada? what was Ichbiah's design, vs. the one that went through? thanks.

Interesting question.

As far as I know, Ichbiah didn't have a design for OO in Ada 95. He wanted the syntax to be something like "class type T is…", as opposed to "type T is tagged…", and he _very_ much wanted "class" to be a reserved word, rather than a mere attribute.

I don't remember Ichbiah proposing any detailed semantics for "class type T is…". His concern, as far as I know, was syntax-oriented.

Of course Ichbiah knew all about OO when he designed Ada 83 — I'm told he was familiar with Simula 67, for ex., which is the Mother of all OOP languages. He left OO out of Ada 83 deliberately. I've no idea whether that was his own choice, or forced by some requirements document like Steelman.

And Ichbiah invented Ada's derived types, upon which Tucker Taft based OO in Ada 95. I've no idea whether Ichbiah had that sort of thing in mind when he invented derived types. But I've been told that derived types were controversial for Ada 83 — the reviewers thought they were useless nonsense, but Ichbiah insisted on keeping them in the language design.

As for other (non-Ichbiah) designs, well I've seen various proposals for "package types", treating Ada packages like "classes" in languages like Simula-67, Java, C++, etc. I'm not sure that works. E.g. the question arises whether types declared in two different "package objects" of the same "package type" should be the same type. I'm not sure it does _not_ work, either. ;-)

[See also 'Computerworld — "The A-Z of Programming Languages: Ada"' in AUJ 29-2 (Jun 2008), p.89. —su]

Size of C & Ada types

From: Adam Beneschan <adam@irvine.com>
Date: Fri, 27 Jun 2008 08:07:57 −0700 (PDT)
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

> Can I use Float instead of Interfaces.C.C_Float?

(…) Only if you know for certain that Float is the same type as the C "float" and don't care about portability. (…)

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Sat, 28 Jun 2008 04:59:38 GMT
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

Interfaces.C should be considered an interface to some specific C compiler, not to C in the abstract. GNAT's version is an interface to gcc C, for example. Whether that compiler is the same as the one used to compile your library, or uses the same representations, is anyone's guess. But on most platforms, most C compilers use similar representations, so you're probably OK using Interfaces.C. If not, then you'd be in the same boat if you used a C compiler.

From: Robert A Duff <duff@adacore.com>
Date: Sat, 28 Jun 2008 13:44:53 −0400
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

> And how the implementation can guarantee it without mandating the representation on the C compiler?

The idea is that the Ada compiler writer chooses a particular implementation of C, and supports interfacing to that. No need to "mandate" — just write the Ada compiler so it mimics what that particular C compiler does. And then the Ada compiler writer writes documentation "this Ada implementation supports interface to the Mumble C compiler, version 1.2.3". The Ada implementation could support interfacing with multiple C implementations, but if the Ada implementation claims to support interfacing to C, it has to support at least one.

> Consider a C compiler that has a switch that selects the representation for fundamental types. It is not uncommon. I can have *the same* program compiled twice by *the same* compiler and the two versions will differ in representation of their fundamental types.

That's conceptually two (or more) C implementations. The Ada compiler writer would document the switches that must be used on the C side. Or maybe there would be similar switches on the Ada side, and the documentation would require them to match.

> How Ada implementation can guarantee anything in this area?

An Ada implementation guarantees that Interfaces.C.C_Float matches the representation of float chosen by a particular C implementation, such as version 1.2.3 of the Mumble C compiler. Not so hard. And quite useful.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 1 Jul 2008 16:10:52 −0500
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

(…) On Janus/Ada, Integer is always 16-bit (for compatibility with our original implementations), but Interfaces.C.Int is whatever the C compiler uses (for most targets, including Windows, that's 32-bit).

If you use Interfaces.C and pragma Convention as intended, your interfacing code ought to be portable — at least with the supported C compiler(s) (and you have a bug to report if it is not). If you use Integer or Float, you're adding portability issues where none are needed.

Indeed, IMHO any code explicitly using the predefined types is wrong in Ada (vis-a-vis portability). And yes, I include the various items in the standard language that do that (including the random number generator and of course type String). It's one of the worst flaws in the language (since it makes string usage not portable, which is idiotic).

From: Peter C. Chapin <pchapin@sover.net>
Date: Fri, 27 Jun 2008 20:56:38 −0400
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

(…) I never assumed that Interfaces.C could somehow magically allow any Ada compiler to work with any C compiler. I'm surprised that some people seem to expect that. However, when I call C code compiled with gcc using GNAT, I would expect Interfaces.C to specify the right types because gcc and GNAT are closely related compilers. Furthermore if I then move my Ada/C program to another Ada/C compiler suite, I would expect it to continue to work (if I'm using Interfaces.C properly, etc, of course).

From: Keith Thompson <kst-u@mib.org>
Date: 28 Jun 2008 14:22:09 −0700
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

> the vendor must provide both the Ada compiler and the C compiler. Are there any vendors out there that don't?

No, the vendor certainly doesn't have to provide both the Ada compiler and the C compiler. The author of the Ada compiler simply has to know how the C compiler represents the various types, in order to get Interfaces.C right. In most cases, the C type representations are mandated, or at least strongly suggested, by the underlying system.

(GNAT/gcc is the only case I know of where an Ada compiler and a C compiler come from the same source.)

From: Robert A Duff <duff@adacore.com>
Subject: Re: Ada array vs C pointer (call by reference)
Date: Sat, 28 Jun 2008 13:52:19 −0400
Newsgroups: comp.lang.ada

> the author of the Interfaces.C implementation has to *know* the representation used by the C compiler, which is, of course, impossible without a crystal ball.

No need for crystal balls. Just read the documentation of the C compiler.

Maybe you're worried that the C compiler will change the size of float in a future version? Well, first of all, that won't happen because it will break existing C programs. But if it does, then that's a new and different implementation of C, and the Ada compiler would have to be modified in order to support interfacing to it. Why is that a problem?

This whole argument started because somebody wanted to use Float instead of C_Float. But there is nothing in the Ada RM saying that Float should correspond to anything in particular. There IS something in the RM saying that C_Float corresponds to float as implemented by the C compiler(s) that the Ada compiler claims to support interfacing to. And the Ada compiler documentation will tell you which C compiler(s) are supported.

From: Robert A Duff <duff@adacore.com>
Date: Mon, 30 Jun 2008 14:55:31 −0400
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

> What do you mean by “the” C compiler?

The one the compiler writer chooses to support, for interfacing.

Or more than one, if the compiler writer so chooses.

>…My impression was that more than one C compiler exists in the world. And even for particular target architectures, more than one C compiler exists for many of these. How is the Ada compiler supposed to know which one you're using, without a crystal ball?

The supported C compiler(s) should be documented. If you use some other C compiler, you're not playing by the rules of the game.

It's the same with interfacing to hardware — if I give you an Ada compiler that generates code for an x86, and you try to run programs on a SPARC, it won't work. That should not be a surprise! ;-)

>…Some people seem to think the Ada compiler will know how the C compiler works, …

Yes, of course it will. It won't know how ALL C compilers in the world work (of course), but it will know about the one (or ones) that are supported.

>… and some seem to go far enough to say that the Ada compiler should be able to *guarantee* that types in Interfaces.C will have the same representation, and that the RM requires this.

Yes, the RM requires this. But you have to obey the Ada compiler's documentation. If it says “compile the C part of your program with gcc version xxx using so-and-so switches”, and use some other C compiler, or some other switches, it might not work.

> I also don't think there's any requirement for Ada to dig around external files to figure out what the representations are. I checked a C book we had lying around, and while it gave no specific definition for the representations of “int”, “float”, etc., it did say that the boundaries of those types are available in <limits.h>. This is (or was) apparently part of the ANSI standard. Does this mean that an Ada compiler has to read <limits.h> [assuming it knows what the default #include directory is] to find information about the types used by the C compiler? I really, really do not think there is any such RM requirement.

The compiler writer can read limits.h, just as well as any other documentation. (…)

From: Tom Moran <tmoran@acm.org>
Date: Sat, 28 Jun 2008 12:49:55 −0500
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

> In any case, there is nothing related to “portability” in Interfaces.C. The types defined there are exactly as non-portable (as far as interfacing is concerned) as any other type in Ada, …
>
> There is of course some added value in Interface.C.XXX — it is documenting the intent.

Vendor supplied libraries with names like “Interfaces.GCC_C” or “Interfaces.Visual_C” would convey that intent just as well, but the source code would require modification to move from, say, Gnat/GCC to some other compiler pair. Interfaces.C would not require source code modification and in that sense is more inter-vendor portable.

From: Robert A Duff <duff@adacore.com>
Date: Fri, 27 Jun 2008 13:00:27 -0400
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

> Interfaces.C.C_Float, which (assuming the Ada implementation gets it right), is guaranteed to match C's float, or of using Float, which has no such guarantee (but saves a little typing). Why would you even consider using Float?

You might have a program that uses Float all over the place, and you want to add some interface to C in one tiny corner of that program. You might be tempted to use Float at the interface to C in order to avoid a lot of type conversions.

It's a pretty good bet that Ada's Float and Ada's Interfaces.C.C_Float, and C's float are all represented the same. If you're using GNAT, I think the documentation guarantees that.

But you're right — the “right” way to interface to C is to use the types in Interfaces.C, and use pragma Convention when you declare your own types.

From: Keith Thompson <kst-u@mib.org>
Date: 27 Jun 2008 11:15:49 −0700
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

And if you take the shortcut of assuming that Float and C_Float are the same, and your program is later compiled on a system where they're not, you're not likely to get an error message. In the worst case, you'll just get subtly wrong answers.

From: Keith Thompson <kst-u@mib.org>
Date: 28 Jun 2008 14:18:03 −0700
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

> A better way to avoid a lot of type conversions is to provide a thin C interface using Interfaces.C.C_Float, and a thin Ada wrapper that does the type conversion to Ada.Float (or some other user provided type).

That will still involve a lot of type conversions at run time, even if you don't have to write a lot of them in your code. If Float and C_Float happen to have the same representation, that's fine. But if they differ, then the conversions could have bad numeric consequences.

To summarize: Programming Is Hard.

From: Rob Norris <rob.norris@baesystems.com>
Date: Thu, 03 Jul 2008 13:44:08 +0100
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

Not too sure where to put my 2p in this thread, but the 'right' way certainly works much better at run time!

We have plenty of code that maps between Ada and C, some of it GNAT and gcc (but different versions on different platforms) and also Ada and C#.

Every so often some one forgets and slips in an Ada float or integer in the data structures. Then at runtime it either get strange results / constraint errors / crashes. Then I come to have a look and go Aha — you should be using Interfaces.C — and these problems go away*

*Except for all the weird alignment issues of records / structures different compilers sometime give.

From: Tom Moran <tmoran@acm.org>
Date: Fri, 27 Jun 2008 13:13:45 −0500
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

(…) Janus Ada 95 Integer is 16 bits (for compatibility with older programs) but Interfaces.C.Int is 32 bits. Another reason to always define your own types (or at least subtypes, so you can change them).

It's amazing to me how many programmers make unwarranted assumptions. In twenty years I suppose all young programmers will assume Integers must "of course" be 64 bits.

From: Adam Beneschan <adam@irvine.com>
Date: Fri, 27 Jun 2008 19:17:08 −0700 (PDT)
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

> (…) the C library (which I am interfacing with) is compiled by unknown C compiler with unknown switches and the author cannot be contacted. The only documentation I have is the C header file.

Got a good disassembler?

If it really isn't documented what type of floats it's expecting, and you have no way to determine what C compiler was being used, you're pretty much down to looking at the disassembled code to figure out how it works, or just trying different possibilities with small data samples where you know what the correct result will be, and trying it with different float sizes until you get the right result. I don't know what sort of different answer you were expecting. Ada is a programming language, not a magician. Nobody here is suggesting that in a case like this, that the Ada compiler should be able to figure out how to interface to your library; and if you think they were, you're overinterpreting.

Furthermore, even if there were some configuration information in the library itself (such as debug information in a symbol table or DWARF section) that gives information on the expected parameters, there is certainly no language requirement that the Ada compiler go delve into the library file to figure this out, and I doubt that any Ada compiler would actually do so. You're on your own, there.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 1 Jul 2008 16:31:36 −0500
Subject: Re: Ada array vs C pointer (call by reference)
Newsgroups: comp.lang.ada

In that case, strictly speaking, you couldn't use the library from C, or Ada, or any other programming language. If you're willing to assume that the code conforms to the normal conventions for the target (that is, works like the "standard" C compiler), then of course Ada (via Interfaces.C) will work the same way.

But clearly, if this is compiled by a "weird" C compiler, you couldn't use it from gcc or MS-C anymore than you could use it from Ada. You could only use it from the "weird" C compiler, and your problem statement says that you don't know what that is. In other words, it is unusable, and you'd be best off rewriting it in Ada (or even a known C).

Nothing Ada-specific about that.

Unconstrained arrays and C

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 28 Aug 2008 16:01:35 −0500
Subject: Re: Possible compiler bug with this simple program
Newsgroups: comp.lang.ada

> In fact, when an Ada subprogram has an unconstrained array parameter with Convention C, it seems to me that the subprogram's body cannot make any use of individual elements of the array, because it doesn't know the index range, so the compiler should reject any indexing of such an array parameter, as well as any attempt to pass it on as a Convention Ada parameter.

This is the subject of AI05-0002-1. (It was carried over from the Ada 95.) [Now, I have to go look this one up because I don't remember anything about what we decided…] Ah, yes:

"We do not require support for C convention interfacing pragmas for unconstrained array objects, unconstrained array function results, and most unconstrained array parameters."

In particular, "An implementation need not support … an Export or Convention pragma applied to a subprogram which has a parameter of an unconstrained array subtype;". The wording goes on to include unconstrained array objects and function results as well.
Note that an implementation *can* support this if it wants; some implementations do implement this with various meanings (Tucker reported that their compiler gives the array maximum bounds) and it was thought to be bad to break user programs that depend on such behaviors. But if it does support it, it ought to do something sensible (raising random exceptions doesn't count). (Also note that it is required to support pragma Import in this case, as C doesn't care about the bounds and they can just be dropped.)

> Conclusion: Your program tries to do something that cannot possibly work, but the compiler should have told you so.

Well, not necessarily (see Tucker's implementation, for instance). But either it should do something defined *or* reject it at compile-time. (Janus/Ada would have rejected the Convention pragma.) In any case, it is not required to support this in any useful way, and, as it is not portable, it should be avoided.

[See also "Size of C & Ada types" in this issue. —su]

Performance of GNAT GPL 2008

From: Alex R. Mosteo <amosteo@unizar.es>
Newsgroups: comp.lang.ada
Subject: GPL 2008 is out… and it's faster!
Date: Thu, 05 Jun 2008 15:33:06 +0200

I'm sure many of you are aware that the GPL 2008 version of GNAT was released yesterday. It includes several goodies, but the first thing I noticed when compiling part of my codebase is that it felt faster.

And sure enough, here is a preliminary test, compiling my particular do-it-all library:

```
$ time gprbuild -Pagpl

GNAT GPL 2007
real    4m35.149s
user    4m1.219s
sys     0m12.529

GNAT GPL 2008
real    3m55.710s
user    3m22.225s
sys     0m12.713s
```

I guess there's been a change in the gcc backend leading to this nice speed-up. Incidentally, I had to change just one line of code to get this (rather large) library to compile (some ambiguity that wasn't perceived as such in the 2007 version).

And now, to test if interfaces are more useable than before…

[See also "GNAT GPL 2008 Edition" in this issue. —su]

Compiler for PIC microcontrollers

From: Britt Snodgrass <britt.snodgrass@gmail.com>
Date: Fri, 18 Jul 2008 20:08:43 −0700 (PDT)

Subject: Re: Is there an Ada port to Microchip PIC?
Newsgroups: comp.lang.ada

> Does anyone know whether there is an Ada port for PICs available? I have do write some code for an 18Fxxx with CAN-bus interface and would like to do it with Ada. I look for native code generation, not for an indirection using intermediate C code.

If you have to use an actual PIC 18Fxxx then I think you won't find an Ada compiler. The 8-bit PICs have a rather ugly programming architecture (especially if writing in PIC assembly language). If you could instead use one of the new PIC-32s which are based on MIPS architecture then its probable you could create or find a GNAT port. There is also a GNAT port for the 8-bit Atmel AVR chips but I don't know if any of them come with a CAN bus capability.

Green Hills Software supports PIC-32 with their C & C++ compilers. I asked if they planned to port their Ada compiler but their response indicated they wanted a first customer to pay for the port.

I've long thought that the lack of Ada compilers for popular, peripheral-rich microcontrollers is the principal reason that Ada asn't been adopted by more embedded programming hobbyists.

[See also “Ada and microcontrollers” in AUJ 28-1 (Mar 2007), pp.32–33. —su]

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

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

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

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

October 09-10 — 2nd **International Symposium on Empirical Software Engineering and Measurement** (ESEM'2008), Kaiserslautern, Germany. Topics include: Reports on the benefits derived from using certain technologies; Empirically-based decision making; Industrial experience in process improvement; Quality measurement and assurance; Evidence-based software engineering; Effort and cost estimation, defect rate and reliability prediction; etc.

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

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

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

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

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

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

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

☺ October 19 7th **"Killer Examples" workshop**. Theme: "Worked Examples" Topics include: (worked) examples of teaching the process of programming, OO modeling and programming, problem solving and programming.

October 19-23 7th **International Conference on Generative Programming and Component Engineering** (GPCE'2008), Nashville, Tennessee, USA. Co-located with OOPSLA'2008. Topics include: Generative techniques for Product-line architectures, Distributed, real-time and embedded systems, Model-driven development and architecture, Safety critical systems; Component-based software engineering (Reuse, distributed middleware, distributed systems, evolution, patterns, development methods, formal methods, etc.); Integration of generative and component-based approaches; Industrial applications; etc.

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

♦ Oct 26-30 2008 ACM **SIGAda Annual International Conference** (SIGAda'2008), Portland, Oregon, USA. Sponsored by ACM SIGAda, in cooperation with SIGAPP, SIGCAS, SIGCSE, SIGPLAN, SIGSOFT, Ada-Europe, and Ada Resource Association (Cooperation approvals pending). Topics include: Transitioning to Ada 2005; Educational challenges for developing reliable, safe, secure software; Ada and SPARK in the classroom and student laboratory; Language selection for a high reliability system; Use of high reliability subsets or profiles such as MISRA C, Ravenscar, SPARK; High reliability standards and their issues; Software process and quality metrics; Analysis, testing, and validation; Use of ASIS for new Ada tool development; Mixed-language development; High-reliability development experience reports; Static analysis of code; Integrating COTS software components; System Architecture & Design; Information Assurance; Ada products certified against Common Criteria / Common Evaluation Methodology; etc. Deadline for early registration: October 22, 2008.

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

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

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

☺ November 10-12 10th **International Symposium on Distributed Objects, Middleware and Applications** (DOA'2008), Monterrey, Mexico. Topics include: Application case studies of distribution technologies; Development methodologies for distributed applications; Interoperability with other technologies; Reliability, fault

	tolerance, quality-of-service, and real time support; Scalability and adaptivity of distributed architectures; Software engineering for distributed middleware systems; etc.
☺ November 10-14	6th IEEE **International Conference on Software Engineering and Formal Methods** (SEFM'2008), Cape Town, South Africa. The aim is to advance the state of the art in formal methods, to scale up their application in software industry and to encourage their integration with practical engineering methods. Topics include: software specification, verification and validation; programming languages and type theory; program analysis; embedded systems; real-time and hybrid systems theory; software architectures and their description languages; light-weight formal methods; CASE tools and tool integration; applications of formal methods and industrial case studies; etc.
November 10-14	19th **International Symposium on Software Reliability Engineering** (ISSRE'2008), Seattle/Redmond, Washington, USA. Topics include: Reliability, availability, and safety of software systems; Validation and verification, testing; Software quality; Software security; Fault tolerance, survivability, and resilience of software systems; Open source software reliability engineering; Supporting tools and automation; Industry best practices; etc.; Empirical studies of any of the above topics.
☺ November 19-20	**Automotive - Safety & Security 2008**, Stuttgart, Germany. Organized by Gesellschaft für Informatik mit den Fachgruppen Ada, etc, and Ada-Deutschland. Topics include (in German): Zuverlässigkeit und Sicherheit für fahrbetriebskritische Software und IT-Systeme; Evaluation und Zertifizierung von Sicherheitseigenschaften automobiler Firmware/Software; Zuverlässige Echtzeit-Betriebssysteme; Fortschritte bei Normen und Standardisierungen; Zuverlässigkeit von Multi-Core-Architekturen; etc.
☺ December 01-04	9th **International Conference on Parallel and Distributed Computing, Applications, and Techniques** (PDCAT'2008), Dunedin, New Zealand. Topics include: Parallel/distributed architectures; Multi-core related technologies; Reliability, and fault-tolerance; Formal methods and programming languages; Software tools and environments; Parallelizing compilers; Component-based and OO Technology; Parallel/distributed algorithms; Task mapping and job scheduling; Security and privacy; etc.
December 01-05	ACM/IFIP/USENIX 9th **International Middleware Conference** (Middleware'2008), Leuven, Belgium. Topics include: design, implementation, deployment, and evaluation of distributed system platforms and architectures for future computing and communication environments. Deadline for early registration: November 4, 2008.
December 03-05	11th IEEE **International Symposium on High Assurance Systems Engineering** (HASE'2008), Nanjing, China. Topics include: Design and development of highly reliable, survivable, secure, safe, and time-assured systems; Policies for reliability, safety, security, integrity, privacy, and confidentiality of high assurance systems; Formal specification, specification validation, testing, and model checking for high assurance systems; High assurance software architecture and design; etc.
December 05-13	4th **International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering** (CISSE'2008), Bridgeport, Connecticut, USA. Topics include: Programming Models and tools, Parallel and Distributed processing, Embedded Systems and Applications, Programming Languages, Object Based Software Engineering, Parallel and Distributed Computing, Real Time Systems, Multiprocessing, etc. Deadline for submissions: October 15, 2008.
☺ December 08-10	14th IEEE **International Conference on Parallel and Distributed Systems** (ICPADS'2008), Melbourne, Australia. Topics include: Parallel and Distributed Applications and Algorithms; Multi-core and Multithreaded Architectures; Resource Management and Scheduling; Dependable and Trustworthy Computing and Systems; Real-Time Systems; etc.
December 09-11	21st **International Conference on Software & Systems Engineering and their Applications** (ICSSEA'2008), Paris, France. Topics include: Components & reusability; Contract-oriented software development; Distributed computing; Evolution (adaptability, maintainability, variability...); Open source issues; Quality control & assurance; Quantitative evaluations (availability, quality, reliability, safety...); Systems engineering and Systems of Systems; Trustworthiness; etc.
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!
☺ December 10-12	6th **International Symposium on Parallel and Distributed Processing with Applications** (ISPA'2008), Sydney, Australia. Topics include: all aspects of Parallel/Distributed Computing and

Networking, and their applications, such as Parallel/distributed system architectures, Tools and environments for software development, Distributed systems and applications, Reliability, fault-tolerance, and security, etc.

December 10-12 **International Conference on Innovation in Software Engineering** (ISE'2008), Vienna, Austria. Topics include: Software engineering, Software components, Reliable software technologies, Maintenance issue, Dependable computing, Reverse engineering, Real-time software, Object-Oriented Programming, Parallel processing systems, etc.

December 17-20 15th IEEE **International Conference on High Performance Computing** (HiPC'2007), Bangalore, India. Topics include: Parallel and Distributed Algorithms, Parallel Languages and Programming Environments, Scheduling, Fault-Tolerant Algorithms and Systems, Scientific/Engineering/Commercial Applications, Embedded Applications, Compiler Technologies for High-Performance Computing, Software Support, etc. Deadline for early registration: November 7, 2008.

2009

January 19-20 ACM SIGPLAN **Workshop on Partial Evaluation and Program Manipulation** (PEPM'2009), Savannah, Georgia, USA. Co-located with POPL'2009 Deadline for submissions: October 12, 2008 (abstracts), October 17, 2008 (papers).

January 24 **2009 International Workshop on Foundations of Object-Oriented Languages** (FOOL'2009), Savannah, Georgia, USA. Following POPL'2009. Topics include: language semantics, type systems, program analysis and verification, concurrent and distributed languages, language-based security issues, etc. Deadline for submissions: October 9, 2008 (abstracts), October 13, 2008 (papers)

February 04-06 **International Symposium on Engineering Secure Software and Systems** (ESSoS'2008), Leuven, Belgium. Topics include: security architecture and design for software and systems; systematic support for security best practices; programming paradigms, models and DLS's for security; program rewriting techniques; processes for the development of secure software and systems; etc. Deadline for submissions: October 24, 2008 (tutorials).

February 16-19 7th **International Conference on Integrated Formal Methods** (IFM'2009), Düsseldorf, Germany.

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

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

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

☺ Mar 08-12 **Track on Object-Oriented Programming Languages and Systems** (OOPS'2009) Topics include: Language design and implementation; Type systems, static analysis, formal methods; Integration with other paradigms; Components and modularity; Distributed, concurrent or parallel systems; Interoperability, versioning and software adaptation; etc.

☺ Mar 08-12 Track on Real-Time Systems (RTS'2009). Topics include: scheduling and schedulability analysis; worst-case execution time analysis; modeling and formal methods; validation techniques; reliability; compiler support; component-based approaches; middleware and distribution technologies; programming languages and operating systems; embedded systems; etc.

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

Analysis and Verification, Program Generation and Transformation, Programming Languages from All Paradigms, etc.

☺ March 16-19 CISIS2009 - **International Workshop on Multi-Core Computing Systems** (MuCoCoS'2009), Fukuoka, Japan. In conjunction with CISIS'2009. Topics include: multi-core embedded systems; programming languages and models; applications for multi-core systems; performance modeling and evaluation of multi-core systems; design space exploration; tool-support for multi-core systems; compilers, runtime and operating systems; etc.

☺ Mar 17-20 12th IEEE **International Symposium on Object/component/service-oriented Real-time distributed Computing** (ISORC'2009), Tokyo, Japan. Topics include: Programming and system engineering (ORC paradigms, languages, RT Corba, UML, model-driven development of high integrity applications, specification, design, verification, validation, testing, maintenance, system of systems, etc.); System software (real-time kernels, middleware support for ORC, extensibility, synchronization, scheduling, fault tolerance, security, etc.); Applications (embedded systems (automotive, avionics, consumer electronics, etc), real-time object-oriented simulations, etc.); System evaluation (timeliness, worst-case execution time, dependability, fault detection and recovery time, etc.). Deadline for paper submissions: October 10, 2008.

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

March 22-29 8th **European Symposium on Programming** (ESOP'2009). Topics include: issues in the specification, design, analysis, and implementation of programming languages and systems, such as Programming paradigms and styles (object-oriented programming, real-time programming languages, etc), Methods and tools to write, reason about, and specify languages and programs (module systems, programming techniques, type systems, program verification, static analysis, language-based security, etc), Methods and tools for implementation, Concurrency and distribution (parallel programming, distributed languages, etc).

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

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

March 24-27 15th **French-speaking Conference on Object-Oriented Languages and Models** (LMO'2008), Nancy, France. Deadline for submissions: October 10, 2008 (papers).

☺ Mar 30-Apr 03 4th **European Conference on Computer Systems** (EuroSys'2009), Nuremberg, Germany. Topics include: All areas of operating systems and distributed systems; Systems aspects of: Dependable computing, Distributed computing, Parallel and concurrent computing, Programming-language support, Real-time and embedded computing, Security, ...; Experience with existing systems; Reproduction or refutation of previous results; Negative results; Early ideas. Deadline for submissions: October 31, 2008 (abstracts), November 7, 2008 (papers).

April 01-04 2nd IEEE **International Conference on Software Testing, Verification and Validation** (ICST'2009), Denver, Colorado. Topics include: Verification & Validation, Quality Assurance, Empirical studies, Embedded and real-time software, Concurrent software, etc. Deadline for submissions: October 3, 2008 (full papers).

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

☺ May 16-24 31st **International Conference on Software Engineering** (ICSE'2009), Vancouver, Canada. Topics include: Specification and Verification; Software Architecture and Design; Patterns and Frameworks; Reverse Engineering, Refactoring, and Evolution; Tools and Environments; Empirical Software Engineering; Development Paradigms and Software Processes; Component-based Software

	Engineering; Model Driven Engineering; Distributed Systems and Middleware; Embedded System; Open Standards and Certification; Software Economics; Dependability (safety, security, reliability); Case Studies and Experience Reports; etc. Deadline for submissions: October 10, 2008 (Software Engineering in Practice), November 2008 (SCORE), November 24, 2008 (research demonstrations), December 5, 2008 (Emerging Results track), December 12, 2008 (doctoral symposium)
☺ May 25-29	23rd IEEE **International Parallel and Distributed Processing Symposium** (IPDPS'2009), Rome, Italy. Topics include: Parallel and distributed algorithms; Applications of parallel and distributed computing; Parallel and distributed software, including parallel programming languages and compilers, runtime systems, fault tolerance, middleware, libraries, scalability, programming environments and tools, etc. Deadline for submissions: October 3, 2008
June 03-06	5th **International Conference on Open Source Systems** (OSS'2009), Skövde, Sweden. Topics include: Software engineering perspectives (F/OSS development environments; Testing, assuring and certifying F/OSS quality and security; F/OSS usability, scalability, maintainability and other quality issues; F/OSS and standards, ...); Emerging perspectives (Licensing, IPR and other legal issues in F/OSS; F/OSS and innovation; ...); Studies of F/OSS deployment (Case studies of F/OSS deployment, migration models, success and failure; F/OSS in vertical domains and the 'secondary' software sector, e.g., automotive, telecommunications, medical devices; F/OSS applications catalog; ...); etc. Deadline for submissions: November 1, 2008
♦ June 08-12	14th **International Conference on Reliable Software Technologies - Ada-Europe'2009**, Brest, France. Sponsored by Ada-Europe, in cooperation with ACM SIGAda Deadline for submissions: December 1, 2008 (papers, tutorials, workshops), January 12, 2009 (industrial presentations).
July 03-08	14th **Annual Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2009), Paris, France.
☺ September 01-04	**International Conference on Parallel Computing 2009** (ParCo'2009), Lyon, France. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments. Deadline for submissions: February 28, 2009 (abstracts), March 31, 2009 (mini-symposia proposals).
September 09-11	8th **International Conference on Software Methodologies, Tools, and Techniques** (SoMeT'2009), Prague, Czech Republic. Topics include: Software methodologies, and tools for robust, reliable, non-fragile software design; Automatic software generation versus reuse, and legacy systems, source code analysis and manipulation; Intelligent software systems design, and software evolution techniques; Software optimization and formal methods for software design; Software security tools and techniques, and related Software Engineering models; Software Engineering models, and formal techniques for software representation, software testing and validation; etc. Deadline for submissions: March 31, 2009 (papers).
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

Association for Computing Machinery

SIGAda 2008 Advance Program
ACM Special Interest Group
on the Ada Programming Language
Annual International Conference
University Place Hotel & Conference Center
Portland, Oregon, USA
October 26-30, 2008

Portland, with Mt. Hood (11,249 feet or 3,429 m)

2008 is an important double anniversary: the Ada 83 Standard was adopted 25 years ago, and the Steelman Report, which gave rise to Ada, was published 30 years

Special Anniversary Keynote Addresses
These exciting speakers are all among our pioneers, having been continuously involved with Ada for at least 25 years.

From Strawman to Ada 2005: a Socio-Technical Retrospective
Dr. Benjamin Brosgol *Senior Technical Staff, AdaCore*

Dr. Brosgol led the Intermetrics "Red" language candidate team in the 1970s, participated in the design of both Ada 83 and Ada 95, and was editor of the Safety and Security Annex of the Ada 95 standard. Under Sun Microsystems' Java Community Process Dr. Brosgol was a member of the Expert Group for JSR-001 (Real-Time Specification for Java, or "RTSJ"), and he is currently a member of the Expert Groups for JSR-282 (RTSJ v1.1) and JSR-302 (Safety-Critical Java Technology).

30 Years after Steelman, Does DoD Still Have a Software Crisis?
Dr. Joyce Tokar *President, Pyrrhus Software*

From 1981-84 Dr. Tokar was responsible for the development of the Gensoft (Western Digital) Ada system. Over the past 20 years, she has been working in the area of mission and safety critical, real-time, and embedded software systems. She is currently active in the area of secure software system development tools and environments, and leads a team in the analysis and evolution of the system of systems software for the US Department of Defense Future Combat System (FCS).

The Ada Paradox(es)
Dr. Jean-Pierre Rosen *President, ADALOG*

Dr. Rosen was a designer of the Ada/Ed system at NYU in the 1980s. After working as a Professor at ENST in France, he formed ADALOG, a training, consultancy, and software development company. Dr. Rosen is Chairman of the AFNOR (French standardization body) group for Ada, and a member of the ARG (Ada Rapporteur Group), the expert group in charge of maintenance and evolution of the language. He was a member of the expert team who controlled the development of the validation suite for Ada 95.

FOR UP-TO-THE-MINUTE TUTORIAL AND SPONSOR/EXHIBIT INFORMATION AND ONLINE REGISTRATION
VISIT http://www.sigada.org/conf/sigada2008 TODAY!

ACM SIGAda Annual International Conference Summary Conference Schedule

Sunday-Monday, October 26-27

Full and Half-Day Tutorials
(see website for details)

Introduction to Ada
Michael Feldman (The George Washington Univ.(ret.))

Languages for Safety-Critical Software: Issues and Assessment
Benjamin Brosgol (AdaCore)

Ada for Real-Time and Parallel Processing
John McCormick (University of Northern Iowa)

Ada for Web Server Development
Ricky E. Sward, The MITRE Corporation

Tuesday, October 28

Greetings from SIGAda and Conference Officers

Keynote Address:
From Strawman to Ada 2005: a Socio-Technical Retrospective
Dr. Benjamin Brosgol
(Senior Technical Staff, AdaCore)

10:30 - 11:00am **Morning Break - Exhibits Open**

Dynamic Analysis of Ada Programs for Comprehension and Quality Measurement
Elaheh Safari-Sharifabadi and Constantinos Constantinides (Concordia University)

re-ADA: Reliable Ada-based Descriptive Architecture for C4ISR via a Quantitative Interoperating Model
Sheldon X. Liang, Lyle A. Reibling and John Betts (Azusa Pacific University)

Sponsor Presentation(s)

12:30 – 2:00pm **Mid-day Break and Exhibits**

A Buffer Container Class Hierarchy Using Ada 2005
Brad Moore

Real-Time Synchronization on Distributed Architecture with Ada 2005
Jim Ras (University of Houston)

Sponsor Presentation(s)

3:30 – 4:00 pm **Afternoon Break & Exhibits**

A Multi-Language Service-Oriented Architecture Using an Enterprise Service Bus
Ricky E. Sward and Kelly J. Whitacre
(The MITRE Corporation)

Ada and Software Engineering Education: One Professor's Experiences
John McCormick (University of Northern Iowa)

Sponsor Presentation(s)

Evening Activities (7:00pm - 10:00pm)
Conference Reception

Wednesday, October 29

Announcements

Keynote Address:
30 Years after Steelman: Does DoD Still Have a Software Crisis?
Dr. Joyce Tokar
(President, Pyrrhus Software)

10:30 – 11:00am **Morning Break and Exhibits**

Implementing the Extended Return Statement for Ada 2005
Tucker Taft (SofCheck, Inc.)

Removing Backward Go-To Statements from Ada Programs
W. Douglas Maurer
(The George Washington University)

Sponsor Presentation(s)

12:30 - 2:00pm **Mid-day Break and Exhibits**

Distributed Status Monitoring and Control using Remote Buffers and Ada 2005
Brad Moore

Anima: A Language for Real-Time Embedded Software Development
Steven Doran

A Distributed, Multi-Language Architecture for Large Unmanned Ground Vehicles
Cynthia Cicalese, Richard Weatherly, Joel Sherrill, Robert Bolling, Kevin Forbes, Robert Grabowski, Keven Ring, and David Seidel
(The MITRE Corporation)

3:30 – 4:00 pm **Afternoon Break**

Workshop:
GNAT: Where Would You Like to See GNAT Go?
Greg Gicca (AdaCore)

Evening Activities (7:00pm - 10:00pm)
Birds-of-a-Feather (BoF) Sessions TBA

Thursday, October 30

Announcements
Ada Europe 2009 Presentation
ACM SIGAda 2009 Presentation
SIGAda Awards

Keynote Address:
The Ada Paradox(es)
Dr. Jean-Pierre Rosen
(President, ADALOG)

Closing Remarks

Ada at FOSDEM 2009
Call for Interest

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

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

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

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

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

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

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

The FOSDEM Team of Ada-Belgium

[1]http://www.fosdem.org
[2]http://www.cs.kuleuven.be/~dirk/ada-belgium
[3]http://www.cs.kuleuven.be/~dirk/ada-belgium/events/06/060226-fosdem.html
[4]http://archive.fosdem.org/2008/call_for_devrooms
[5]http://www.fosdem.org/2009/brainstorm
[6]http://listserv.cc.kuleuven.be/archives/adafosdem.html

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

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

Conference Chair

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

Program Co-Chairs

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

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

Tutorial Chair

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

Exhibition Chair

Pierre Dissaux
Ellidiss Technologies
Pierre.Dissaux@ellidiss.com

Publicity Chair

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

Local Chairs

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

In cooperation with
ACM SIGAda

General Information

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

Schedule

01 December 2008	Submission of regular papers, tutorial and workshop proposals
12 January 2009	Submission of industrial presentation proposals
09 February 2009	Notification to all authors
09 March 2009	Camera-ready of regular papers required
11 May 2009	Industrial presentations, tutorial and workshop material required
8-12 June 2009	Conference

Topics

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

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

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

Program Committee

Alejandro Alonso, Universidad Politécnica de Madrid, Spain
Leemon Baird, US Air Force Academy, USA
Johann Blieberger, Technische Universität Wien, Austria
Maarten Boasson, University of Amsterdam, The Netherlands
Bernd Burgstaller, Yonsei University, Korea
Dirk Craeynest, Aubay Belgium & K.U.Leuven, Belgium
Alfons Crespo, Universidad Politécnica de Valencia, Spain
Juan A. De la Puente, Universidad Politécnica de Madrid, Spain
Raymond Devillers, Université Libre de Bruxelles, Belgium
Michael González Harbour, Universidad de Cantabria, Spain
Javier Gutiérrez José, Universidad de Cantabria, Spain
Philippe Dhaussy, ENSIETA/LISyC, France
Andrew Hately, Eurocontrol CRDS, Hungary
Jérôme Hugues, Telecom Paris, France
Günter Hommel, Technischen Univesität Berlin, Germany
Hubert Keller, Institut für Angewandte Informatik, Germany
Yvon Kermarrec, Télécom Bretagne, France
Fabrice Kordon, Université Pierre & Marie Curie, France
Albert Llemosí, Universitat de les Illes Balears, Spain
Franco Mazzanti, ISTI-CNR Pisa, Italy
John McCormick, University of Northern Iowa, USA
Stephen Michell, Maurya Software, Canada
Javier Miranda, Universidad Las Palmas de Gran Canaria, Spain
Scott Moody, Boeing, USA
Daniel Moldt, University of Hamburg, Germany
Laurent Pautet, Telecom Paris, France
Laure Petrucci, LIPN, Université Paris 13, France
Luís Miguel Pinho, Polytechnic Institute of Porto, Portugal
Erhard Plödereder, Universität Stuttgart, Germany
Real Jorge, Universidad Politécnica de Valencia, Spain
Alexander Romanovsky, University of Newcastle upon Tyne, UK
Jean-Pierre Rosen, Adalog, France
Lionel Seinturier, Université de Lille, France
Frank Singhoff, UBO/LISyC, France
Oleg Sokolsky, University of Pennsylvania, USA
Ricky Sward, MITRE, USA
Tullio Vardanega, Università di Padova, Italy
Francois Vernadat, LAAS-CNRS, Université de Toulouse, Insa
Andy Wellings, University of York, UK
Jürgen Winkler, Friedrich-Schiller-Universität, Germany
Luigi Zaffalon, University of Applied Sciences, W. Switzerland

Industrial Committee

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

Call for Regular Papers

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

Proceedings

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

Awards

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

Call for Industrial Presentations

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

Call for Tutorials

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

Call for Workshops

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

Call for Exhibitions

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

Grants for Students

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

13th International Real-Time Ada Workshop

17-19 April 2007
Woodstock, Vermont
USA

Sessions:
Ada and Other Standards
Conclusions and Plans for next IRTAW

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: Ada and Other Standards

Chair: Ben Brosgol

Rapporteur: Mario Aldea

1. Session Goals

The main goals of this session were to:

- consider whether a new binding between Ada and POSIX is needed/desirable
- look at current efforts to update POSIX and "Real-Time Java" in terms of impact on or "lessons learned" for Ada

2. POSIX Ada binding

Stephen Michell presented his paper on "Interfacing Ada to Operating Systems" [1]. He stated that the POSIX Ada Binding has not been updated since 1998, even though POSIX has undergone two significant revisions and the new Ada 2005 has been approved. Taking into account those changes, the current binding is mostly correct from the functional point of view, but:

- there are some minor errors/inconsistencies
- it does not include the new functionality added to POSIX during the last few years
- some functionality included in the binding is now directly supported in Annex D of Ada 2005

Stephen also stated there are other operating systems that do not follow the POSIX standard and are important nowadays, like the Windows family or other operating systems for embedded platforms.

Instead of adapting the current POSIX Ada binding to the new situation, Stephen proposed to update it to a general interface to operating systems, implemented in a set of packages which would be children of Ada.Interfaces.

2.1 Discussions

Some general points about the POSIX Ada binding philosophy were noted:

- The general idea of the binding is to provide interfaces only for the functionality not provided by the Ada language
- One of the main objectives of the binding is to allow Ada tasks and POSIX threads to interoperate, for instance by being able to share data using mutexes, or to synchronize through condition variables. Jose Ruiz mentioned that the GNAT compiler allows POSIX threads to call Ada protected objects (the run-time library registers them as foreign threads)
- The difference between the priority band concept in Ada 2005 and the way scheduling policies are defined by POSIX was identified as a potential difficulty.

The following concerns about the utilization of the binding were raised:

- The general feeling was that the binding has had very few users. Many programmers just create a small binding for the few functions they actually use in each particular application. At this point some doubts appeared about whether it is worthy to put some effort on a new binding. There was consensus on at least putting some effort to correct the minor errors/inconsistencies.

One of the minor details that should be corrected is the fact that the current POSIX/Ada binding references the old POSIX standards instead of the new ones.

It was pointed out that the current binding is a mixture between Ada and C styles and it would be desirable to develop a more Ada-like binding. Some people agreed on that but it was questioned if it is worthy for the Ada community to put such a big effort in this project.

There was a suggestion of doing a minimum change in the binding in order to mark as obsolete those services currently included in the binding that have an equivalent in the Ada 2005 standard.

2.2 Conclusions

The main conclusions of this part of the session were:

- There was an agreement that future editions of this workshop could help in the technical decisions related with the new POSIX/Ada binding, but that the workshop itself should not take over the responsibility of the revision of the binding
- The workshop participants agreed that, at least, a minimum update of the binding is desirable in order to correct minor errors/inconsistencies and wrong references to POSIX standards.

3 Real-Time Java

Ben Brosgol made a presentation about the history and current status of Real-time Java, including a summary of the main RTSJ (Real-Time Specification for Java) features and a list of the existing implementations.

Nowadays the main effort in the RT Java area is the Safety-Critical RT Java, officially named "Java Specification Request 302" (JSR-302). Two members of the IRTAW workshop are included in the "Expert Group" in charge of developing this project: Andy Wellings and Ben Brosgol. JSR-302 progress is slow since there are two competing proposals: HIJA (High-Integrity Java Applications) and

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

Aonix with its "Scalable Real-Time Java proposal".

Ben pointed out several aspects of RT Java that could be interesting for Ada:

- Real-time programming paradigms
  - Periodic, aperiodic, sporadic threads
- General capabilities
  - Annotations to guide static analysis
- Interesting functionality
  - Asynchronous Event Handling:
    - Application control over "fire count" to deal with bursts
    - Pass data when an event is fired
  - General mix of Priority Inheritance and Priority Ceiling Emulation
  - Ability to awaken a suspended thread via synchronous exception
- "Exotasks" (from IBM, based on Giotto)
  - Not threads but periodic code that is dispatched based on a scheduler
  - Dedicated heap that is garbage collected
  - No shared memory; communication via deep copy over typed ports
  - Eclipse-based environment allowing annotated timing constraints

3.1 Discussions

The subsequent discussion were centered on the following issues:

- Including garbage collection in Ada: it could be interesting for some applications that use unchecked deallocation. It is noticed that nothing in Ada prevents from using a garbage collector, so maybe this is not actually an issue.
- Different kinds of "physical" memory: the ability to place data in a specific kind of memory (fast access memory, flash memory, ...). This service is provided by RTSJ and POSIX but not by Ada.
- Mix of Priority Inheritance and Priority Ceiling Emulation: can be a source of problems due to the nested locks, but it was pointed out that priority inheritance is an interesting functionality for large systems built from "components" developed independently.
- Awaken a suspended thread: a possible alternative in Ada consists of blocking the task in a protected entry.

3.2 Conclusions

The main conclusions of this part of the session were:

- The Real-time Ada community should keep an eye on the current efforts on RT Java and check if its advances could be interesting for future revisions of the Ada standard
- The workshop encourages the Real-Time Ada community to continue research on topics like garbage collection, different kinds of memory, or priority inheritance in Ada.

References

[1] Michell, S. *Interfacing Ada to Operating Systems*. (this issue).

Interfacing Ada to Operating Systems

Stephen Michell

Maurya Software Inc, Ottawa, Ontario, Canada; email: stephen.michell@maurya.on.ca

Abstract

This paper examines approaches used by the Ada programming language to interface to explicit operating system services such as events and sockets. We examine the potential for updating a specific interface, the POSIX Ada binding, to a more gereral interface to operating systems services. An approach is proposed to unify classes of services such as synchronous and asynchronous file IO in such an interface.

1 Introduction

The Ada Programming Language has always had a deficiency of interfaces to operating systems. Except for the simple interfaces such as File_IO, most provided interfaces (by the OS) to get services such as Messages, Queues, Events and Asynchronous_IO are not available as Ada packages and subprograms.

A notable exception to this was the POSIX Ada binding [APOSIX], which began life as IEEE 1003.5c, and was adapted to Ada 1995 and published as IS14519:2001. Even this interface shows its age as POSIX has moved from a notion of Processes to one that includes Processes and Threads. Ada has added Protected Objects, access to subprograms and protected subprograms, and Interfaces. Collectively, these changes to Ada provide better ways to consider aiding capabilities to an interface such as an interface to an operating system.

We examine the POSIX Ada Binding, and then propose a set of interfaces to generalized operating systems to replace the POSIX Ada Binding, and finally show how new Ada capabilities could be used to integrate diverse concepts such as File_IO and Asynchronous_File_IO.

This paper is organized as follows. Section 2 dis cusses the layout of the POSIX Ada Interface, and the types of services that it manages. Section 3 proposes a new set of interfaces to operating systems. Section 4 proposes a way to incorporate synchronous and asynchronous behaviours when using operating system calls. Section 5 gives conclusions.

2 The POSIX Ada Binding

The POSIX Ada Binding [APOSIX98] began in the 1980's as an interface to the most popular operating system specification [POSIX]. By its nature, it captured the interface as a set of packages and subprograms to express the functionality. [APOSIX] was updated to incorporate changes to POSIX and Ada and published as IEEE 1003.5c:1998 and IS14519:2001. Since then, the binding has not been updated, even though POSIX has undergone 2 significant revisions, incorporated much of POSIX Real Time, and Ada has incorporated major functionality changes for Ada 2005. Some of the changes undertaken by Ada have a direct impact on the binding by duplicating functionality originally supplied by POSIX with Ada-supplied primitives.

In [WMM2006] we did an analysis on the existing binding and POSIX and recommended that WG9 begin a revision of the binding.

The POSIX Binding to Ada can be characterized as a set of Ada packages that define subprogram calls to the underlying OS to implement file IO, environment variables, memory management, process scheduling, timers, events, signals, semaphores, and sockets. A summary of the interface packages is in Table 1.

There has always been a significant tension between POSIX and the Ada runtime system with regards to the management of computational resources such as task scheduling, intertask notifications, and memory. The benefit of using POSIX was that it provided capabilities (different thread scheduling paradigms, timers, and events) that were not available natively in Ada, and that it provided intertask(thread) services to programs that do not have the rich concurrency paradigm of Ada. The challenge has always been that there are significant issues trying to send or receive POSIX signals or events between tasks and POSIX thread, either within a single program or in separate programs. It has almost always been the case that one should not intermix Ada tasks with POSIX threads.

Nevertheless, the POSIX Binding to Ada has been successfully used to interface Ada programs to POSIX. Now that Ada has been updated and has added many of the tasking paradigms that were provided by POSIX, it is reasonable to see if this binding could be replaced by a set of interfaces as part of Ada that are less OS-specific and which better integrate with the Ada runtime.

4 Ada and OS services

We note that [POSIX01] is just one of the operating system interfaces needed in a modern Ada program. There is a set of operating systems called Windows [Win95], [Win98], [Win2000,] [Win] that are exceedingly important to Ada programs, as well as embedded systems such as [Wind River]. Argueably, it does not make sense for Ada to attempt to focus its energies on the mapping to a single class of operating systems; rather a strong case can be made

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

to bring the appropriate OS services that have been defined for POSIX, extend them to general purpose OS's and include them in a set of packages in Ada.Interfaces. For example, all functionality related to memory should be in Ada.Interfaces.Memory, such as in Ada.Interfaces. Memory.Locking and in Ada.Interfaces.Memory.Maps.

We therefore propose that instead of updating the POSIX Binding to Ada, that a new set of energies be created to replace them which also handle the general issue of interfacing to the underlying operating system.

Table 2 presents a proposed set of package interfaces to general purpose operating systems. These interfaces roughly correspond to the interfaces from [APOSIX98] but would be recasted to make better use of Ada05 characteristics and lessons learned, such as use of protected operations to specify timing_events as described in [Ada05] section D.15. An example of one such interface is shown in figure 1.

Table 1 POSIX Binding capabilities in Ada

Capability	Binding Package	Ada Capability	Comments
Page_Alignment	POSIX_Page_Alignment	Nothing	
Environment Varia	POSIX_Process_Environment	Ada.Environment_Variables	
Time Zone	POSIX_Calendar	Ada.Calendar	
Process Primitives	POSIX_Process_Primitives	Nothing, tasks = threads	
Events	POSIX_Signals	Exceptions, entries, prot subprograms and entries	Some thread oriented, others process oriented
Per-thread-timing	POSIX_Process_Times	Ada.Execution_Time	
Process identificati	POSIX_Process_Identification	Ada.Task_ID, Ada.Task_Attributes	
Time	POSIX_Calendar	Ada.Calendar	
File-access	POSIX_permissions		
System Config	POSIX_Configurable_System_Limits	Nothing	
File manipulation	POSIX_Files	Ada.Directories	
	POSIX_File_Status	Ada.Directories	Incomplete
File_IO	POSIX_IO	Ada.Text_IO, Ada.Streams_IO	Ada missing notions of locking and ownership
File Locking	POSIX_File_Locking	Nothing	
Asynchronous_IO	POSIX_Asynchronous_IO	Nothing	
Terminal_IO	POSIX_Terminal_FunctionS	Nothing	
User Information	POSIX_User_Database	Nothing	
Group Information	POSIX_Group_Database	Nothing	
Synchronization	POSIX_Semaphores	Task Entries & POs	
Mutexes	POSIX_Mutexes	Task Entries & POs	
Condition variables	POSIX_Condition_Variables	Task Entries & POs	can be dynamic- POSIX
Memory Locking	POSIX_Memory_Locking	Nothing	
	POSIX_Memory_Range_Locking	Nothing	
Memory_Mapping	POSIX_Memory_Mapping	Nothing	
Shared Memory	POSIX_Shared_Memory_Objects,POSIX_Generic_Shared_Memory	Partitions	
Scheduling	POSIX_Process_Scheduling	Ada Real Time Dispatching	
Timers	POSIX_Timers	Ada.Calendar, Ada.Real_Time.Timing_Events	
Messages	POSIX_Message_Queues	Nothing for process, entries for tasks&POs	
Task Management	Ada's Task_ID		
Interprocess comn	POSIX_XTI	Annex E	
Sockets	POSIX_Sockets	Nothing	
Events	POSIX_Events	Nothing	

Table 2 Proposed Ada.Interfaces structure for OS Interfacing

Capability	Ada Capability	Ada.Interfaces
Page_Alignment	Nothing	
Environment Variables	Ada.Environment_Variables	
Time Zone	Ada.Calendar	
Process Primitives	Nothing, tasks = threads	
Events	Exceptions, task entries, protected subprograms and entries	
Per-thread-timing	Ada.Execution_Time	
Process identification	Ada.Task_ID, Ada.Task_Attributes	
Time	Ada.Calendar	
File-access		
System Configuration	Nothing	
File manipulation	Ada.Directories	
	Ada.Directories	
File_Limits		
File_IO	Ada.Sequential_IO,	
Asynchronous_IO	Ada.Direst_IO, Ada.Text_IO, Ada.Streams_IO	File.IO see section ???
File Locking	Nothing	File.Locking
Terminal_IO	Nothing	File.Terminal_IO
User Information	Nothing	Users
Group Information	Nothing	Groups
Data Interchange	Nil	
Synchronization	Task Entries and prot objects	Semaphores
Mutexes	Task Entries and prot objects	Mutexes
Condition variables	Task Entries and prot objects	Condition_Variables
Memory Locking	Nothing	Memory.Locking Memory.Range_Locking
Memory_Mapping	Nothing	Memory.Maps
Shared Memory	Partitions	Memory.Shared_ Memory.Generic_Shared_Memory
Scheduling	Ada Real Time Dispatching	Scheduling.Process_Scheduling Scheduling.Thread_Scheduling
Timers	Ada.Calendar, Ada.Real_Time.Timing_Events	
Messages	Nothing for processes, entries for tasks, protected objects	Message_Queues
Interprocess commun	Annex E	XTI
Sockets	Nothing	Sockets
Events	Nothing	Events

```
package Ada.Interfaces.File_IO is

  type File_Type is interface;
  procedure Put(File : File_Type;...);
  ...
  type Asynchronous_Notify is access protected procedure F;
  type Asynchronous_File_Type is interface;
  -- put, get, etc inherited from File_Type
  - - or extended to add capabilities such as
  - - including an accessprotectedprocedure that
  - - is called when the action completes
  procedure Put ( File : Asynchronous_File_Type;
                  IO_Complete : Asynchronous_Notify; ...);
  procedure Wait_For_Completion( File : Asynchronous_File_Type;...);
end Ada.Interfaces.File_IO
```

Figure 1 Example IO interface using Interface types

3 Asynch Task Interfaces

A challenge for applications interfacing Ada programs to operating system services is that these services often provide explicit thread control in the OS while Ada tasking (equivalent to threads) is managed by the language. The challenge for an Ada-OS interface that this view can be quite different since Ada has protected operations, entries, and conditional and timed entries are based on the state of the called task while most OS's use Events, Semaphores, Signals, One-way messages and sockets. Integrating the two views means having to match OS ID's and states and Ada ID's and states explicitly.

For example, OS signals raised by a task or are caught by a task can be synchronous or asynchronous. Ada95 provided mechanisms to notify tasks about events, the protected procedure, and protected entry, but these are asynchronous and are insufficient to map the OS services directly to the language.

Now that Ada 2005 has added access to protected procedures, there are more opportunities to interfaced to OS services than previous versions. Instead of "magic" subprograms that handle an interface, Ada2005 permits us to bridge the concurrency divide between the OS and Ada tasks at protected subprogram boundres. For every signal, event, file_IO, message or socket that requires a concurency-based interface, use protected interfaces and access to protected procedures to capture these events from outside the Ada program and to bring them into the Ada program. Examples of this are provided in the Ada 2005 Rationale [BARNES2006].

Figure 1 shows an outline of a sample package that could integrate synchronous IO and asynchronous IO as part of an interface to operating systems. The obvious parallels can be drawn for Sockets, Events and Message_Queues where a single interface can be provided and both the synchronous interface and a parallel asynchronous interface.

We propose that such interfaces be used in the definition of a collection of services to bind Ada programs to Operating systems. For example, a task could call Ada.Interfaces. File_IO.Put on File_Type of Asynchronous_IO and return immediately, then call either Wait_For_Completion or call the protected operation that was passed to Put for the parameter IO_Complete. By setting up multiple protected interfaces, a single task can manage multiple asynchronous read or write operations, without the need to explicitly task ID's, into POSIX thread ID's. This approach is usable for many of the POSIX interfaces.

4 Conclusion

This paper has shown an approach to update the POSIX-Ada binding to a general interface to operating systems. It proposes to add these as children of Ada.Interfaces. It also shows an approach that could unify the treatment of synchronous interfaces and asynchronous interfaces using the new Ada interface mechanism.

Bibliography

[Ada83] ANSI 1815:1983, The Ada Programming Language

[Ada95] ISO/IEC 8652:1995, The Ada Programming Language

[Ada05] ISO/IEC 8652:2007, The Ada Programming Language

[APOSIX] IEEE 103.5c:1998 and IS14519:2001, POSIX Binding to the Ada Programming Langauge

[BARNES2006] Barnes, John, Rationale for Ada 2005, available online from www.adaic.com

[POSIX96] IEEE 1003.1,2,3, and IS9945-1:1996, The Portable Operating System Interface

[POSIX2003] IEEE 1003.1,2,3, and IS9945-1:2003, The Portable Operating System Interface

[WMM2007] Wong, Luke, Michell, Stephen, Moore, Brad, *Initial Work Scope Summary for updating Ada POSIX Binding IS 143519:2001 to the Ada Programming Language IS8652:2007*, available from ISO/IEC/JTC1/SC22/WG9 Ada Working Group as document N477r.

Session: Conclusions and Plans for next IRTAW

Chair: Juan A. de la Puente

Rapporteur: Santiago Urueña

1 Introduction:

The final session summarized the main conclusions of the workshop and addressed some open issues. Another objective was to decide whether a new IRTAW is needed, and, if so, the location and time frame.

2 Language Issues

The conclusions of the first session were summarized. Alan Burns stated that an Ada Issue should be created and submitted to the ARG to correct the current definition of the EDF protocol because the standard is wrong [1]. The proposal was accepted unanimously by the 18 participants, and he will be in charge to write the AI.

With respect the omission of the requeue statement in object oriented programming [2], the consensus was that the requeue is a useful primitive in combination with interfaces, and that Ada should be consistent in this point. Alan Burns reminded that the ARG explicitly asked the workshop to study this problem, and the participants approved (17 yes, 0 no, 1 abstention) to further investigate this topic, that a static solution (a pragma) is an effective mechanism but the implementation costs must be investigated.

Finally, the workshop agreed that a Ravenscar profile for distributed systems [3] is very interesting, and that research should proceed on it. First the requirements should be defined and then the restrictions should be developed.

3 Programming Patterns and Libraries

Andy Wellings said that a set of real-time programming patterns [4, 6] for Ada is interesting, and that the work will continue in a future workshop thanks to the ARTIST project. The University of York will hold a full-day meeting in October. The addition of servers [5] to the framework will also be discussed in the meeting.

Jorge Real asked when the source code will be available, but Andy Wellings replied that currently there is not a full Ada 2005 implementation, so the patterns must be first tested.

4 Implementation Experience with Ada 2005

The rapporteur of the third session [7, 8] summarized that the conclusions were first to investigate a better means of execution-time accounting, including a better model for interrupt handling; to reaffirm the workshop support of user-defined scheduling; and that there is not a consensus on whether those execution-time timing mechanisms should be added to Ravenscar.

Juan A. de la Puente then presented a slide to further explain this last open issue, explaining the use envisaged: a task can have a static execution-time timer which is armed in each activation. If in a rare event the task consumes more CPU time than its assigned WCET the handler of the execution-time timer will awake a monitoring task. This monitoring task can make a system-dependent recuperation procedure like mode change or safe stop, avoiding the faulty task to disrupt other tasks.

He reminded that this was discussed in past IRTAWs, and it had the support of the workshop at that time. Andy Wellings said that there is no need to add CPU timers because a monitoring task can check the correct behaviour of the rest. Tullio Vardanega then stated that CPU timers are needed to allow multiple Ravenscar partitions to coexist. In the opinion of Alan Burns, that cannot be considered a change to Ravenscar, but a new profile. The final consensus was that the workshop encouraged further investigation on this topic.

5 Beyond Ada 2005

Jorge Real, as the chair of the fourth session, expressed that there was not enough time to finish the discussion, but that the workshop reached some consensuses. The first one was that Ada needs more standardized support for multiprocessor systems, like the ability to specify the affinity of a task to specific processors [9]. Andy Wellings proposed to continue the discussion in the next meeting, and to set up a consortium for Ada and the upcoming hardware architectures. Michael González Harbour and Juan A. de la Puente were also interested.

Then the workshop continued with the topic about stream-based parallel systems [10]. Neil Audsley stated that this is a potential future direction for Ada, as there is no competitor. For example, the C language is not as strong as Ada with respect to the memory footprint and energy savings. Finally, Juan Zamorano said that the proposal made by Santiago Urueña is a candidate for the distribution model of a future Distributed Ravenscar. Tullio Vardanega suggested that this will be discussed in the next workshop, and Santiago Urueña expressed that he will investigate this issue.

6 Ada and Other Standards

Mario Aldea started the session talking about the discussion about the bindings to POSIX [11]. The consensus was that

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

the standard should be updated with the minimum number of changes. Stephen Michell expressed that anyone interested in participating can contact him by e-mail and subscribe to the mailing list.

This was followed by a summary and discussion about RTSJ, and Juan A. de la Puente stated that the Ada real-time community should continue following the progress of other languages like Real-Time Java, opening new proposals to include in Ada. Ben Brosgol suggested several candidates for consideration including garbage collection, mixed Priority Inheritance, and the Priority Ceiling protocol.

7 Future Plans

Alan Burns proposed to post the session reports on the web page of the workshop, so the ARTIST web site can link to them. Stephen Michell further suggested to have a permanent IRTAW web site so the pages of all workshops are always available. Ben Brosgol agreed to have this permanent web site on the web site of the Ada Resource Association.

Finally, it was discussed whether another IRTAW should be held in the future. Tullio Vardanega felt that it would be desirable, and Jorge Real stated that there are a lot of open issues. The unanimous decision was that another workshop is needed and should be planned for approximately 18 months from now, namely in September 2008. Tullio Vardanega said that he would be happy to hold the next IRTAW in Italy, while Neil Audsley volunteered to be the Program Chair.

References

[1] Zerzelidis, A., Burns, A., Wellings, A.J. *Correcting the EDF protocol in Ada 2005.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

[2] Wellings, A.J., Burns, A. *Integrating OOP and Tasking — The missing requeue.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

[3] Urueña, S., Zamorano, J. *Building High-Integrity Distributed Systems with Ravenscar Restrictions.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

[4] Wellings,A.J., Burns, A. *A Framework for Real-Time Utilities for Ada 2005.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

[5] Burns, A., Wellings, A.J. *Programming Execution-Time Servers in Ada 2005.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

[6] Pulido, J., de la Puente, J.A., Bordin, M., Vardanega, T., Hugues, J. *Ada 2005 Code Patterns for Metamodel-Based Code Generation.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

[7] Urueña, S., Pulido, J., Redondo, J., Zamorano, J. *Implementing the New Ada 2005 Real-Time Features on a Bare Board Kernel.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

[8] Aldea, M., González Harbour, M. *Operating System Suport for Execution Time Budgets for Thread Groups.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

[9] Burns, A., Wellings, A.J. *Beyond Ada 2005: Allocating Tasks to Processors in SMP Systems.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

[10] Ward, M., Audsley, N.C. *Suggestions for Stream Based Parallel Systems in Ada.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

[11] Michell, S. *Interfacing Ada to Operating Systems.* In ACM Ada Letters, Volume XXVII, Number 2, August 2007.

13th International Conference on Reliable Software Technologies

Ada-Europe 2008

Venice, Italy,
June 16-20, 2008

Session
“Ada and Software Engineering Education”

Session Report: Ada and Software Engineering Education

Chair: Jorge Real

Rapporteur: Luís Miguel Pinho

Abstract

At the Ada-Europe 2008 conference, a special session was devoted to the subject of Ada in Software Engineering Education. This short paper reports on the proceedings of the session.

Keywords: Ada, Software Engineering, Education

1 Introduction

As the Conference Organizer openly put to the Ada community, the objectives of this session were to discuss (quoting from [1]):

- what languages should be used in the software engineering curriculum in comparison to what are actually taught: where Ada stands in that business and how Ada could instead serve that curriculum;
- what it takes (or should take) for a graduate to learn Ada in comparison to the (false) perception that if Ada is not taught in the curriculum then "there are no Ada programmers around".

There were around twenty attendees in the session, from eight countries, from both academia and industry, and from both sides of the Atlantic.

2 Education Requirements and Issues

The first part of the session included four presentations, providing a variety of experiences on the subject issue:

- Ed Schonberg, as a professor at the New York University, presented an analysis of the use of Java as the first programming language, and argued that Ada and C++ are superior for introductory Computer Science courses [2];
- John McCormick, of the University of Northern Iowa, presented the experience of students implementing software to control a railroad model (a large multi-tasking real-time embedded system), and the gains observed when switching from C to Ada [3];
- Jean Pierre Rosen, of Adalog, presented an assessment of the importance of education per se, not only for Ada, but for software engineering in general [4]; and
- Carl Brandon, of the Vermont Technical College, described the use of Ada to implement the software for a pico-satellite (CubeSat) [5].

In this section we provide a brief summary of the presentations (thus biased by the focus of the presenter – and the synthesis capacity of the listeners). Following in this Journal issue the reader will retrieve the papers derived from (or that led to) the presentations at the session.

2.1 Java considered harmful

This presentation was based on the paper that appeared in the Crosstalk Journal [6], and that stirred an intense online discussion on various threads (both pro and against the authors' perspectives).

Ed Schonberg started out by arguing that the quality of education in the software engineering field is visibly decreasing. And, although not the root cause of it, the Java language does not seem to help. Ed Schonberg also noted that Java is often not being targeted as a programming language, but rather as a tool to learn computer science. Particularly important, Java was chosen as the target of Ed's talk because it has become the central piece of elementary computer science education.

The presenter also noted that it is important to provide programmers with the ability to zoom across levels of abstraction: from the small, such as code performance, to the very large, such as software frameworks.

Concerning programming in the small, the presentation outlined different topics were Java hinders the understanding of code performance, such as the virtual machine, garbage collection and object-orientation (whose perils were also highlighted).

Afterwards, the presentation focused on two areas that Java has a serious impact on: composition and concurrency. In the former, Java forces composition by delegation, leading to the proliferation of different, multi referenced, objects. In the latter, the concurrency model of Java is very low level, not providing the effective abstractions that have been devised in the field in the last 30 years.

The talk also outlined some concerns on the program-structuring tools of Java. Starting from the noting that Java only provides the class as a structure building block, an important conclusion was that some of the basic notions of software engineering are hard to describe in Java.

As for frameworks, the introduction of the term 'bloat' as a technical characterization concerning Java, was proposed as a consequence of the proliferation (i.e, bloating) of classes and objects imperceptibly to the programmer.

Ed Schonberg also presented an interesting parallelism between program correctness and psychoanalysis. This was used to lead to the use of compilers as theorem provers, and the strengths of Ada's type system for this.

Finally, a challenge was made to the audience. By noting that programming is more remodelling than building houses, Ed noted that language readability is extremely important. It is therefore important to make widely available Ada-based introductory computer science texts.

2.2 Developing software to control a railroad model

In the second presentation of the session, John McCormick started by introducing the context of the Real-Time Embedded Systems Course in the University of Northern Iowa. In this course, students relate fundamental scientific real-time computing issues with practical software development.

The presentation focused on the practical project that students must complete, where in 15 weeks they must develop a real control system with around 10-15 KSLOCs. Instead of using simulators, which do not cause the problems associated with actual systems and do not provide the real experience required by those hiring computer science graduates, students must develop software to control a real large railroad system model.

This railroad system was selected, and is actually being used, not only because the equipment is affordable and available, but as also because it provides for versatile challenges (both discrete and continuous control systems). Also important, this system highly motivates students into the course and into computer science at the University.

After describing the project's platform, a set of minimum, high-level requirements is given to the students, including terms such as no train collisions, fault recovery, and the participation of at least one human operator, including the interference of a "devious" professor.

Afterwards, John showed the results of the experience study. During the first six years of the project, C was used for the control software development. An important point was made that, even with an increasingly amount of supplied code (up to 60%), not a single group was able to fulfil the minimum requirements.

Assuming that the problem was in the tasking model of C (or rather the lack of it), a shift was then made from C to Ada, as Ada had a much higher level of tasking abstraction.

As a consequence of this shift, in the first year 50% of the groups finished their projects. After 7 years, around 75% of the groups regularly complete their projects, and only with around 10% of supplied code.

However, by analysing the students' logbooks, it was possible to determine that the major problem was not C's low level tasking model, but Ada's modelling of scalar quantities. Ada's type system "caught" most of the common errors early in the development, where they are more easily fixed.

As a conclusion, the programmers should spend as much effort designing the scalar types as usually done with the modelling of larger classes in object-oriented systems.

2.3 The importance of education

In the third presentation, Jean-Pierre Rosen shared his experience regarding the importance of education, which he personally gathered in several Ada-related projects.

His presentation started by antagonizing the thesis "Ada Education is important" and its antithesis "Ada Education is not important".

As to the former, Jean-Pierre pointed out that the main problem is that people are literally thrown into Ada projects without Ada knowledge, on the grounds that Ada is a very readable language. His opinion is that as bad or insufficient Ada education is actively harmful to Ada.

He then presented some examples:

- Not knowing the language is harmful. The programmer would in fact rather claim that the problem is with the language and not with his/her knowledge of it.
- It is possible to program in Ada as in C. This incurs poor usage that makes it difficult to program and causes the programmer to fight the compiler.
- Not understanding Ada typing will make the compiler an enemy. If unchecked conversion is used as a simple solution, then there more problems than benefits will ensue.

In conclusion, Ada is the problem!

As for the antithesis, Jean-Pierre noted that many software developers do not have education in computer science. The basic CS education is no longer taught and design methodologies are out of fashion.

Therefore, his question is why we focus on the language, as many other more important skills are missing. As an example, he noted that it is simpler to teach Ada to someone that knows secure systems than the opposite.

Also, it is necessary that we do not give the impression that only experts can use Ada, as it draws programmers away from Ada.

Making the synthesis of his opinion, he proposed the good news that the language design is right: the lack of features is not the reason industry goes away from the language.

However, as bad news, the problem is not Ada. It is the lack of software engineers, with education in computer science.

Jean-Pierre then made some proposals of what can be done to change this situation. Ideally we should teach people how to think: the goals and principles of software engineering, algorithms and problem solving, and Ada in the remaining time.

Programming should be taught as an engineering discipline, with its own set of rules, which should be understood and followed by the programmer. Finally, Ada should also be taught to those already active in Ada projects. Software Engineering education should also be promoted.

The conclusion of the presentation is that Ada adoption would be the natural consequence of teaching Software Engineering.

2.4 Developing software for a pico satellite

The final presentation of the session was by Carl Brandon, of the Vermont Technical College (VTC). The presentation shared the author's experience with a student project for implementing the software for a pico satellite.

The platform (CubeSat) is a custom built satellite platform, based on the Texas Instruments MSP430 micro controller, with (target) 116 KB of Flash and 8 KB of RAM.

The same platform will be used in the Arctic Sea Ice Buoy system being developed which will also reuse some of the software components. The hardware and software for these systems will be developed by VTC students in their senior year projects.

Although no Ada complier exists for this platform, Ada was chosen for its reliability, as the software is considered to be mission critical. Simultaneously, the project can be used to teach Ada.

SPARK is also used, considering the integrity requirements of the software being developed. There is no way to fix the software after launch, which thus entails a very high cost of failure both in dollars and in time. Another motivating factor in the project was the possibility for Software Engineering majors to work in a high-integrity real project.

The Arctic Sea Ice Buoy will use sequential SPARK, whilst RavenSPARK is foreseen for the CubeSat, even though Carl noted that concurrency in the CubeSat is only in the attitude control, and if avoidable it will actually not be used.

Since there is no Ada compiler for this particular platform, the solution found was to translate Ada code (after going through SPARK) to C, using Ada Magic, and afterwards using a C compiler and the Salvo Real-Time Operating System. Students can therefore write the source code in Ada and SPARK and be able to produce code for the MSP430 in one go.

3 Discussion

After the presentations, the floor was open to questions and comments from the audience. The discussion started by analysing what is the current situation with the programming language used for the first CS courses.

Several examples were provided of Java being used, although John McCormick provided a counter example of switching to Java in the first two courses, and after 5 years, considering it a failure, switching back to a mix of Ada and C++.

Another argument was then put forward by Theodor Tempelmeier, suggesting that the shift to Java was also being industry-driven. This argument was reinforced by Ed Schonberg noting that computer science is no longer attractive. The market is becoming split into two different groups. On one side, the software architects, highly paid, and on the other side, the programmers. The focus of industry is on quantity of low-pay programmers, for whom knowing Java is sufficient. And there is nothing that Educators can do about this.

Jean-Pierre Rosen reinforced this by noting that the current trend is to have simple jobs, which are easy to outsource.

The audience also debated as to whether computer science students were decreasing. John presented his case in that in the USA, the number of students was down to 60% of the numbers 8 years ago. Jean-Pierre provided his opinion of France in that there is a decline of 50% in the number of students. Theodor noted that in Germany there is a decline, but also a major shift from technical computer science to economic/management computer science.

Tullio Vardanega then provided Italy's example, where computer science is still more attractive than engineering. He does not consider that there is a crisis in science, but thinks that courses must provide students with the necessary tools (languages) for them to be able to point out in industry the best solution for each specific case. Industry also wants people able to offer good answers to difficult and pressing questions.

Ed Schonberg then asked if the world is interested in Reliable Software. In his opinion, it is not Ada that matters per se, but that reliable software does. That is the idea that the Ada community should put forward. However, as Ed pointed out, if society puts up with faulty software, then nothing can be done for general-purpose software. But for safety applications, then it does matter. Jean-Pierre noted that the automotive industry is a possibility for a more widespread use of software reliability.

Tullio then noted that he has not encountered yet an industry that is not concerned about reliability. However, industry is pragmatic. In a particular case, one company switched from Ada to C because they were afraid that Ada would not last (contrary to their expectations of C). He thinks that the automotive industry has the same expectations. Therefore, the demand for reliability is there. It is however uneducated.

Tullio also presented another example, where an industry, with totally misplaced expectations, went trough a very difficult effort to shift from Ada to automatically-generated C, hoping for lower costs in a small percentage of the total.

Jorge Real offered his view on the conflict between industry and academia. In his view, the argument behind the shift to Java is popularity, which is deemed to be equivalent to finding a job. And what is popular currently is Web development. That is why Java is popular. Society accepts and even expects computers to fail. You can always switch off and on again. The question should be how to persuade society that computers can and should be reliable and that software could be reliable too.

John drew a parallel with the automotive industry in the 60's, where cars where expected to fail. Then the Japanese automotive industry started to build more reliable cars and the rest of the sector had to follow.

Julio Medina then asked for a poll. At what level is reliability taught? His opinion is that reliability is often taught at the master level, which is a problem. Ed agreed with it, adding that he believes that in many courses there is no reliability at all. Programmers believe that they never fail, and everyone takes for granted that programming is simple. However, we need to spread the word that programming (the discipline not the language) is complex, but at the risk of chasing students away. We currently do the opposite: we try to persuade students that it is simple.

The audience then analysed the consequences of the higher level of abstraction in programming. Julio argued that programmers should see all layers, from assembly to the high-level. According to Julio, programmers nowadays abstract from the underlying infrastructure, and believe that any "mistake" will be solved by some hidden features underneath.

The same problem was pointed out by several participants. Robert Dewar added that it is a current trend for students not needing to know hardware, assembly or compilers. He noted that NYU went from 2 to 1 course for assembly, C and architecture issues.

Ed Schonberg then noted that Web and real-time development are two disjoint areas. The latter will continue to exist, and it is this area that the Ada community addresses. It would be good to also address the former, but it is not the main priority.

4 Conclusions

The time was up before we were able to draw up a synthesis of the discussion. The session chair however volunteered a brief summary, considering that an important move would be to push forward software reliability in the curricula of Computer Science and related courses. He noted that this effort could best fit the framework of ACM computing curricula.

References

[1] Vardanega T., email to the GAP list, November 2007

[2] Schonberg, E., Dewar, R. *A Principled Approach to Software Engineering Education or: Java Considered Harmful*. (this issue)

[3] McCormick, J. W., *Ada and Software Engineering Education: One Professor's Experiences*. (this issue)

[4] Rosen, J.-P. *Is Ada Education Important?*. (this issue)

[5] Brandon, C. *Use of Ada in a Student CubeSat Project*. (this issue)

[6] Dewar, R. Schonberg, E., *Computer Science Education: Where Are the Software Engineers of Tomorrow?*, CrossTalk, The Journal of Defense Software Engineering, January 2008. Available at http://www.stsc.hill.af.mil/CrossTalk/2008/01/ 0801DewarSchonberg.html

A Principled Approach to Software Engineering Education, or Java considered Harmful

Edmund Schonberg, Robert Dewar

Adacore Inc, 104 5th Avenue, NYC 10011, USA; email: Schonberg@gnat.com, dewar@gnat.com

Abstract

We examine the use of Java as a first programming language, in the light of well-established principles of software engineering, and the increasing concern with correctness, performance, and maintainability. We argue that Java is markedly inferior to Ada or C++ as a language for introductory Computer Science courses, and that its widespread use in the training of tomorrow's software engineers is counterproductive.

Keywords: Software Engineering, Education, Java, Ada.

1 Introduction

It is a well-established fact (first discussed by E.Dijkstra) that the programming language in which programmers receive their first instruction has a large impact on their programming habits. Current instruction in Computer Science (see for example ACM's Computing Curricula 2005 [1]) minimizes the teaching of multiple programming languages, which makes the impact of the first language even more critical. Java is more and more the language of choice for introductory programming courses. We argue that this is a poor choice.: we examine the drawbacks of Java as a teaching language under four headings, which following Koolhas et al are conveniently labelled small, medium, large, and extra large [4]. Before delving into the details, let us establish the limits of our arguments:

a) We consider that programming will retain a central role in all software construction: there is no automatic programming machinery in sight that will make programming a secondary activity.

b) Programming remains a demanding intellectual discipline. The separation between "designers" and "programmers" attempts to create a hierarchy of skills (and salaries!) but this separation is artificial and counterproductive: software authors (to coin a term) must have a rigorous training that includes solid foundations in software engineering. We are particularly concerned with safety- and security-critical systems, that present considerable engineering challenges.

c) We do not debate the importance of Java in today's software industry, and do not discuss the merits of the language in its industrial and commercial applications: our concern is with the training of software engineers.

d) One of the fundamental skills of a good software engineer is the ability to zoom, that is to say to change the focus of his activity from the very large (software architecture) to the very small (efficiency of generated code, cost of synchronization, etc.). The education he receives must develop this ability, and the languages in which he is taught plays a vital role in this. This argues for the use of a wide-spectrum language from the beginning.

2 Programming in the small

This is the realm of algorithmic analysis: the programmer must be able to estimate reliably the performance of code, in terms of time and space. The disadvantages of Java in this respect are several:

a) The Java virtual machine hides the real architecture.. The JVM is basically a simple stack machine, which makes it easy to port, but it includes some complex operations whose cost will vary from target to target. The use of just-in-time compiling to speed up critical paths makes the performance of a Java program even harder to estimate. It is certainly the case that for many applications (in particular Web programming) a casual approach to performance is acceptable. For safety-critical systems and real-time systems this is not sufficient.

b) Most critically, garbage-collection adds a hard-to-quantify cost to Java programs. Furthermore, the presence of the garbage collector encourages what we might call a profligate style of programming, where objects are created freely for the simplest of computations. For example, object-oriented methodologies encourage the "boxing' of atomic values (transforming an int to an Integer object, for example). So as to honour the concept that "everything is an object". As a result, the simplest computation will involve the dynamic creation of heap-allocated objects, and it will become impossible to estimate the time behaviour of code. This attempt at unification is inspired by Smalltalk, but it is interesting to note that Eiffel abandoned this unified model early in its design [5] and that C++, like Ada, sensibly maintains a clear distinction between elementary values and composite ones. The difficulties of analyzing the performance of large Java systems is vividly described in [7].

3 Programming in the medium

This is the realm of abstraction and encapsulation. In Java (and to a large extent in C++) the fundamental concept is the class, which we must contrast with the various type constructors in Ada. We include in this category the primitives for concurrent programming.

It is well-known that when designing new abstractions (software components) composition is more important, and used more often than inheritance. Yet in Java composition can only be obtained by delegation, that is to say by embedding a pointer to an object inside of another one. By contrast, in Ada (and to some extent in C++) composition is obtained through aggregation, subtyping, and unions (free in C++, discriminated in Ada). As a result Java design leads to a proliferation of objects, heavy use of dynamic storage, and structures that pointer-heavy and therefore wasteful of storage. The impact of this on performance is well described by Mitchell et all [6]..

3.1 Concurrent Programming

Concurrency is an aspect of Java that is decidedly low-level:

a) Synchronization is per-method, and there is no direct thread-to-thread communication, except through shared memory.

b) Distributed locking operations make it harder to formalize concurrent behaviour, and the suspend/resume mechanisms are notoriously error-prone (race conditions, deadlock).

c) The semantics of priorities and the queueing regime are not defined precisely enough to guarantee real-time behaviour.

Concurrency is much better taught with tasks and protected objects, as presented in [3]: standard concurrent paradigms (producer-consumer, mailboxes, semaphores, broadcasting, etc.) are easily constructed with them. Finally, the use of the Ravenscar profile (part of the Ada 2005 standard) allows the construction of concurrent programs with fully deterministic behaviour.

4 Programming in the large

The only program-structuring mechanism of Java is the class. This is the most glaring deficiency of Java from the point of view of software engineering: there is no proper separation between specification and implementation, and there is no mechanism for hierarchical composition of components.

a) The separation between specification and implementation is not just a matter of information hiding (which is handled by public/private dictions in all languages of interest): it is of the greatest importance in the simultaneous development of large systems. In Ada, design starts with package specifications. Once these are agreed upon, development of client code can proceed independently of the implementation of these specifications. Finally, the separation between specification and body simplifies incremental recompilation: a client need not be recompiled when changes in the implementation of a package do not affect its specification.

b) The class is too small a unit out of which to design systems, but there is no grouping mechanism that allows the semantically coherent aggregation of classes. The Java notion of a package is more akin to that of a library of weakly related components. In contrast, the Ada package provides a mechanism for type aggregation, a visible dependency graph through context clauses, and a flexible model of system extensibility through child units.

c) Java cannot deal with subtyping independently of inheritance: there is a conceptual confusion between subtyping as enrichment (the usual notion of inheritance0 and subtyping as subsetting. This is a problem with all O-O methodologies, and is not just a philosophical issue, but one with pedagogical import (see e.g. the discussions around the circle-ellipse relation: which should be considered a subtype of the other?[8]).

After the concept of package, the most important contribution of Ada to software engineering is the notion of constraint (including constraints on scalar types). This notion has no analogue in other languages. Constraints are of course a simple but powerful example of program assertions: they define behaviour more precisely, and they can be checked statically byt the compiler, or enforced dynamically. In either case they pin down the semantics of the program in ways that are not available in other languages.

5 Programming in the very large

Most large software systems today combine components that are themselves aggregates (subsystems) consisting of a number of packages or classes, often written in different languages. For the most part the mechanisms for assembling these components are embedded in an Interactive Development Environment (IDE), of which Eclipse is a well-known ecxample. At this level it would appear that the choice of language plays a smaller role, but there are two areas in which Ada presents definite advantages: interfacing with other languages, and static program analysis.

a) Ada formalizes the description of components that may be written in other languages, by means of pragmas (Import, Export, and Convention). The Ada library provides data conversion routines to trasnsform e.g. Ada self-describing strings into C zero-terminated strings. These pragmas and library routines allow the Ada compiler to verify the type coherence of a program that has foreign language components. By contrast, the JNI mechanism in Java, and the mechanisms provided by other languages, lose most type checking in the presence of components written in other languages.

b) Larger and faster machines make whole-program analysis possible over programs with tens and hundreds of thousands of source lines. This makes it possible to detect programming defects at compile time (uninitialized variables, race conditions, constraint violations, etc) that are beyond the reach of unit by unit compilation. However, the power of static analysis, depends on the richness of information available to the analyzer, and to a large extent this depends on the richness of the type system of the language. In this context it is useful to think of a compiler as simple theorem prover: every diagnosed error is a proof that a certain invariant is violated somewhere. As with any deductive system, the richer the set of axioms, the more interesting the proofs that can be derived from it. In this sense, redundancy within the program text is beneficial, because it makes it possible to check for consistency. Programmers often regard Ada as too verbose, and balk at the substantial declarative machinery that they have to use, but these declarations are precisely what makes Ada compilers so much more precise in their diagnostics.

There is a continuum between type checking as performed by a compiler, ambitious static analysis as performed by a tool such as Softcheck's Inspector [7], and program verification as obtained with Spark [2]. However, what makes Inspector and Spark possible (and what makes the error messages of a good Ada compiler so precise) is the strong static typing model of Ada. No other language today has a typing system that is rich enough to support such tools. The quality of diagnostics produced by these tools is particularly valuable for beginners, and is a revelation for programmers coming from other languages.

6 Conclusions

We can summarize the shortcomings of Java as an introductory programming language as follows:

a) Java hinders the understanding of code performance.

b) Java design methodologies lead to a proliferation of objects, heavy use of dynamic storage, and data-structures that are pointer-heavy and thus wasteful.

c) The Java model of concurrency low level and error-prone, and the garbage-collected environemt prevents its use in real-time applications.

d) The fundamental separation between specification and implementation is absent in Java, hampering good software engineering development practices.

e) Without the notion of constraint, Java has no way of specifying useful invariants to describe program behaviour.

Some of these deficiencies also apply to C++ as an introductory language, even though as a wide-spectrum language is does satisfy the concern with performance analysis described in section 3. There is no further need to enumerate the reasons for the superiority of Ada over either Java or C++ as an introductory programming language.

References.

[1] *ACM Computing Curricula 2005*. At http://www.acm.org/education/curric_vols, 2006.

[2] John Barnes: *High Integrity Software: The Spark Approach to software Safety and Integrity*. Addison-Wesley, 2003.

[3] Alan Burns, Andy Wellings. *Concurrent and Real-time Programmming in Ada 2005*, Cambridge University Press, 2006

[4] Rem Koolhas et al, *S M L XL*, Monacelli Press, 1997

[5] Bertrand Meyer, *Object-Oriented software Construction*, Prentice Hall, 1997.

[6] Nick Mitchell, Gary Sevitsky, Harini Srinivasan. *The diary of a Datum: an approach to analyzing runtime complexity in Framework-based applications,* in Workshop on Library-centric software design, OOPSLA 2005

[7] Softcheck. *Inspector,* http://en.wikipedia.org/wiki/SofCheck_Inspector

[8] Wikipedia. *Circle-ellipse problem*, in http://en.wikipedia.org/wiki/Circle-ellipse_problem

Ada and Software Engineering Education: One Professor's Experiences

John W. McCormick

University of Northern Iowa, Cedar Falls, IA 50614, USA.; email: mccormick@cs.uni.edu

Abstract

How do you select a programming language for your project? Few developers have the luxury of coding the same design in multiple languages to compare language merits. For over twenty years my undergraduate students have implemented the same large (10-15K lines), multi-tasking, real-time embedded system. In one 15 week semester, student teams specify, design, and implement software to control a substantial model railroad layout.

Students implement everything from device drivers for custom I/O hardware to high-level decision making algorithms. Student teams have implemented the project in both Ada and C. This paper describes the course, the laboratory, the project, and an analysis of the results achieved with each of the implementation languages.

Keywords: education, real-time, laboratory, Ada, C, model trains

1 Introduction

This paper describes a course, *Real-Time Embedded Systems*, offered by the Department of Computer Science at the University of Northern Iowa. *Real-Time Embedded Systems* is a capstone course taken in the third or fourth year of study. To perform well in this course, students must integrate knowledge from their previous work in computer science, electronics, English, mathematics, and physics. Students are exposed to the fundamental scientific issues in real-time computing [1] and gain practical skills of software development. A major goal is to educate software engineers capable of working as members of an interdisciplinary development team. Many topics are covered at a survey level. For example, students in the course learn just enough of the basic concepts of control theory to be able to communicate with a control engineer and to implement a simple control algorithm. Graduates of the course have gone on to work in a wide range of domains including avionics, communications, manufacturing, transportation, farm machinery, and medical instrumentation.

2 Acknowledgment

This paper is based on an earlier work *We've Been Working On The Railroad: A Laboratory For Real-Time Embedded Systems*, Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, St. Louis, Missouri, USA, pp 530-534 © ACM, 2005. http://doi.acm.org/10.1145/1047344.1047510.

3 Laboratory

An introductory undergraduate course in real-time embedded software development should acquaint students with the fundamental scientific issues of real-time computing and practical skills in embedded software development. While the theoretical issues can be covered without a laboratory, real-time embedded software development skills require the experiences that a laboratory provides. A major problem is finding equipment suitable for teaching these skills.

Simulators are commonly used to give students experience with real-time embedded systems programming [2]. Typically these simulators do not provide many of the frustrating problems associated with physical systems. In most embedded systems projects, hardware and software are developed in parallel. Gathering evidence for the determination of whether a fault is in the hardware or the software is an important skill for the embedded systems programmer. Lack of experience with real systems is a major reason cited by engineers who would exclude computer science graduates from their development teams.

I have used a computer-controlled model railroad in my real-time embedded systems course for more than 20 years. Some of the advantages of using a model railroad in the laboratory are:

- Model railroad equipment is readily available and priced well below typical laboratory equipment.
- Model railroads provide a wealth of problems from both the discrete and continuous real-time domains.
- Undergraduate computer science students easily understand the electronics and physics.
- Students are highly enthusiastic about writing software to control a model train layout.

4 Model railroad equipment

The model railroad I use is HO scale (1:87). While smaller scales would permit more equipment in the laboratory, they are more expensive, more difficult to maintain, and less readily available.

To run multiple trains on their layouts, model railroaders traditionally divide the track into electrically isolated sections called blocks. Modellers use many toggle and rotary switches to connect a particular power supply (called a cab) to a group of track blocks beneath each train. In my layout, the computer controls the power and polarity applied to each of 40 blocks. Today's model railroad enthusiasts often use more modern direct digital control of locomotives to solve the problem of multiple train control. In these systems, the track is used as a communications bus. Commands are addressed to specific locomotives and broadcast on the track. I have rejected this approach as it provides fewer software development problems and less experience with analogue electronics.

Turnouts (commonly called switches) are controlled by gear- and screw-driven switch machines. As they are electromechanical devices, turnouts sometimes fail to move to the desired position. A stuck turnout can often be freed by moving it back and forth.

Engineers drive a train via a hand-held control unit, a small box with two buttons, a knob, and two toggle switches. There is no hardware for debouncing the buttons or switches. Typical student projects assign knobs for train throttles, buttons for whistles and brakes, and toggle switches for train direction (forward or reverse) and for setting the next turnout ahead of the train (left or right).

In order to do closed loop control, it is necessary to obtain feedback on the process being controlled. For the model train this feedback consists of the trains' locations as a function of time. On my layout, this information is obtained from 51 Hall effect sensors installed on the track. These sensors are triggered by small magnets attached to the front of every locomotive and to the rear of each caboose. Sensors simply report the presence of a magnet; there is no information provided as to which magnet triggered it.

Sound provides another dimension to the layout. We have hardware that generates sounds from digital recordings of diesel locomotive engines, air brakes, air horns, and bells. An independent ASCII text to speech processor provides feedback for locomotive engineers and is a useful debugging tool for the student software engineers. Sending debugging output to the speech processor rather than a display screen allows students to observe the behaviour of the hardware anywhere in the laboratory while receiving verbal information on the state of their software.

5 Computing hardware

A number of different hardware configurations have been used over the long history of this project. In my first laboratory, students developed their control software on a Digital Equipment Corporation PDP 11/24. They used a serial link to download executable programs to a PDP 11/23 computer. In 1989 I received a laboratory improvement grant from the National Science Foundation (NSF) enabling me to replace the PDP 11/24 with a microVAX II and the PDP 11/23 with an rtVAX (optimized for real-time). My current system uses inexpensive PCs for both development and target machines. Future plans include using a variety of traditional embedded systems processors.

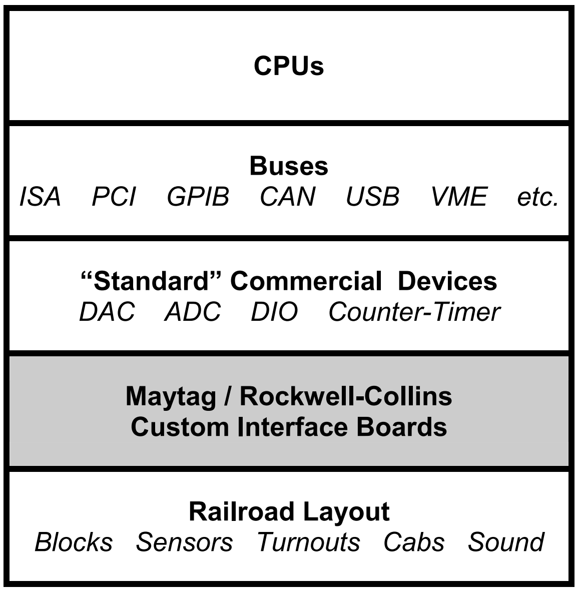
CPUs
Buses
ISA PCI GPIB CAN USB VME etc.
"Standard" Commercial Devices
DAC ADC DIO Counter-Timer
Maytag / Rockwell-Collins
Custom Interface Boards
Railroad Layout
Blocks Sensors Turnouts Cabs Sound

Figure 1. System architecture

5.1 System architecture

The interface hardware connects the control computers to the railroad hardware. Figure 1 is a diagram showing the layers within a model railroad control system.

One or more CPUs are connected to commercial analogue-to-digital converters (ADC), digital-to-analogue converters (DAC), TTL level digital I/O (DIO), and counter-timers. The connections may be made through any of a number of different buses. I use custom hardware to connect these devices to the railroad layout. In the past, this interface layer was hand-built on wire wrapped and soldered prototyping boards.

As a direct result of presentation and publication of previous work [3, 4, 5], nearly 70 schools and corporate training departments have requested detailed specifications of the laboratory. Most were discouraged by the large amount of effort (500-plus hours) required to assemble the necessary interface electronics. With the support of Maytag Corporation and Rockwell-Collins, I have designed and manufactured circuit boards that make this aspect of building the laboratory much easier for those wishing to duplicate my efforts. The interface hardware consists of four major subsystems (block control, turnout control, train sensors, and sound) detailed in the next sections and two minor subsystems (cabs and hand-held controllers). The cost of the interface electronics for a small layout is about $1,200. Interface costs for my large layout were $2,900.

5.2 Block control subsystem

The block control subsystem controls the power applied to each block of track in the railroad layout. Figure 2 shows a single track block circuit. Each Maytag / Rockwell-Collins block circuit board contains 12 such circuits.

The two analogue outputs of the block control circuit are connected to the rails of a block of track to supply power to the train on that block. Each circuit has four digital inputs and eight analogue inputs. Three of the digital inputs (cab

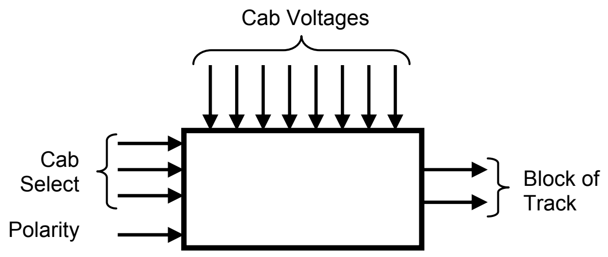

Figure 2 Block control circuit

select in Figure 2.) are used to select which one of the eight analogue inputs will be used to power the track block – an analogue version of the multiplexer students study in computer organization. The remaining digital input is used to select the polarity of the voltage applied to the track. The polarity controls the direction the train moves on the block. The analogue inputs (cab voltages in Figure 2) may be supplied by either digital-to-analogue converters or by programmable counter-timers. The latter provide a pulse width modulated signal for controlling the speed of a train. Pulse width modulation gives more precise control of the DC motors in the locomotives than is possible with simple voltage level control.

5.3 Turnout control subsystem

The turnout control subsystem is designed to control Tortoise™ brand switch machines. Other brands can be used. Figure 3 shows a single turnout control circuit. Each Maytag / Rockwell-Collins turnout circuit board contains 24 such circuits.

The two analogue outputs of the turnout motor control circuit are connected to a switch machine motor. A series of contacts on the switch machine provide an analogue signal (sense in Figure 3) that indicates whether or not the linkage has reached the end of its travel.

A single digital input (direction in Figure 3) selects the direction to turn the switch machine motor. Because switch machines take two to three seconds to change, a turnout has four possible states: left, right, moving left, and moving right. Rather than use two output lines to determine the state of the turnout, we use the desired direction (the input value) in combination with an output value (in position in Figure 3) that reports whether the switch machine motor has reached the desired direction.

5.4 Train detection subsystem

The train detection subsystem connects the Hall effect sensors on the track to a digital input/output (DIO) board with interrupt capabilities. It may also be used with other TTL level sensors. Each Maytag / Rockwell-Collins sensor circuit board handles 64 sensors. LED's are provided to aid in debugging interrupt handlers.

The Hall effect sensors are located on the boundaries between track blocks. When a locomotive is detected, the software must power up the next block before the wheels

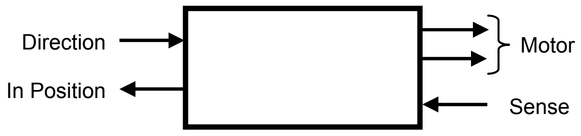

Figure 3 Turnout control circuit

bridge the gap between blocks. This is a hard real-time deadline. Failing to power the next block in time will blow the block power supply fuse. Students prefer to test their software with locomotives rather than by pushing cars with plastic wheels that cannot bridge the electrical gap between blocks. I supply each team with five fuses. Usually, within a week after teams begin testing their block control software, every hardware and auto part store in town is sold out of 4 amp fuses.

5.5 Sound subsystem

The sound subsystem provides a digital interface to up to four Dallee LocoMatic™ railroad sound units [6]. Each unit provides diesel engine sounds that vary in proportion to throttle settings, air brake release sounds, air horn

6 Laboratory assignments

The four credit-hour course has three 50-minute lectures and a two hour laboratory session each week. The early laboratory sessions are used to review (or learn) and practice with the features of the implementation language that are important for the completion of their project. These features include data types and structures, control structures, modules and packages, input/output, classes and objects, concurrent programming, and exceptions. Later laboratory sessions are devoted to developing code that will be directly applied to their projects, including polling and interrupt-based device drivers, implementation of a whistle class, and implementation of a turnout class. As mentioned earlier, turnouts are electromechanical devices that sometimes fail to operate correctly. The software must detect and correct turnout failures. Students derive their turnout driver code from state machines they develop to model turnout behaviour.

7 Course project

Students work in teams of three or four to complete a substantial (10K - 15K lines) project. Each team writes all the code necessary (from low level device drivers to high level control logic) to produce a bootable image. Teams are free to formulate their own projects. Minimum project requirements include:

- Running multiple trains.
- At least one train controlled by a human engineer.
- No train collisions.
- Detecting and recovering from hardware failures, such as turnouts, sensors, lost cars, and devious professors.

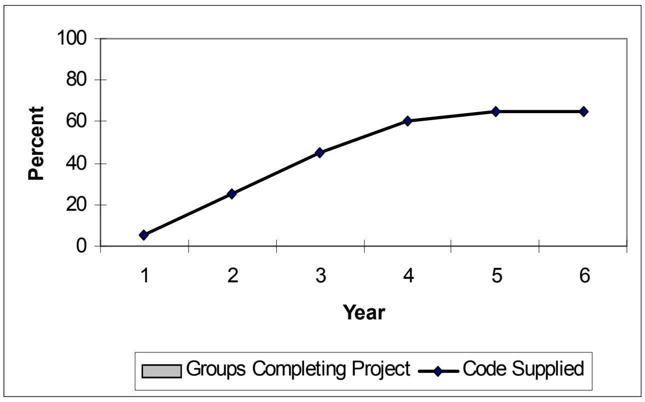

Figure 4 C language project completion rate (zero) and amount of code supplied by instructor

Over the years, train races, train wars, and dynamic scheduling problems have been the most popular project themes. Many teams also implement additional safety features such as throttle limits and protection for locomotive transmissions.

Deliverables for the project include:

- A system concept document.
- A detailed user's manual.
- Object modelling documents.
- Compiled class specifications.
- Unit (class) test plans.

These deliverables are used as milestones throughout the course to help ensure that students keep up with the demanding schedule necessary to complete the project. One of my major tasks is to work with teams on their systems concept document to reduce overly optimistic proposals into ones that can be completed. Students are aware of the completion rates of past teams (presented later in this paper) so they understand that they can complete the project by the end of the semester.

Students also keep an engineering notebook with a detailed record of their individual and team activities. They maintain a separate time log that I review and sign once a week.

Student teams do exhaustive module testing where behaviour of a particular object (such as a turnout or locomotive) is well understood. Integration testing is bounded by the amount of time at the end of the semester. All test plans are approved and test results certified by a member of the team selected as the team's test manager.

8 Programming languages

During the first six years that the real-time systems course was offered, students developed their railroad control code in C. As shown in Figure 4, no team successfully implemented the minimum project requirements when the C language was used. To ease student and teacher frustrations, I made an increasing amount of my solutions available to the teams. Figure 4 shows that even when I provided nearly 60 percent of the project code, no team was successful in implementing the minimum requirements.

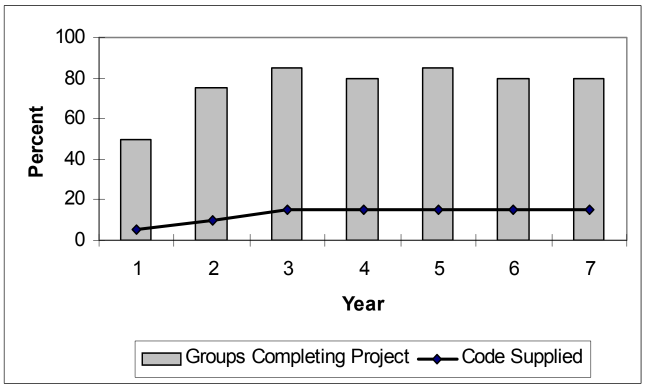

Figure 5 Ada language project completion rate and amount of code supplied by instructor

Along with the new hardware, provided by the NSF funding in 1989, was a collection of DEC compilers. Thinking that the low level of tasking provided through semaphores was the major contributor to the problem of incomplete projects, I selected a language with a much higher level of tasking operations – Ada. Ada was designed for implementing complex real-time embedded systems. Ada's rendezvous and protected objects provide easy to use, safe mechanisms for task communication and synchronization. For example, the three semaphores needed to protect and synchronize a typical bounded buffer in C are replaced by a single protected object with enqueue and dequeue operations. These high level operations reduce the chance of typical errors (such as omitting a semaphore operation) while providing as good or better execution times.

I expected a disaster the first year with the new equipment and new language. As in a real-life embedded systems project, I was building the hardware while my students were writing the software. I finished the hardware with only four weeks remaining in the semester. But to my amazement, nearly 50 percent of the student teams had their projects working before the end of the semester. I had supplied them with only two sample device drivers. As shown in Figure 5, when I supplied some additional software components (simple packages not relevant to the real-time aspect of the project), more than 75 percent of the student teams routinely completed their projects. These completion rates have remained steady through the present time.

Through an analysis of the students' engineering notebooks, I found my original hypothesis, that the major problem was C's low-level tasking mechanism, to be incorrect. While Ada's high level of abstraction for tasking did reduce development time for the students, it was the accurate modelling of scalar quantities (integers, real numbers, and enumeration values) that contributed the most to Ada's success in this course. Hours spent locating a C function call with two swapped int parameters was reduced to a quick fix of a syntax error in the equivalent Ada code.

Similarly, the calculation of an out of range track block number that took a team a full day to find in their C program is detected and reported immediately by the Ada run-time system. This conclusion is consistent with studies done on the nature of wicked bugs in software [7, 8] where over 80 percent of the programming errors in the C/C++ programs studied were a result of problems with scalars; problems that do not exist in Ada.

9 Ada development environment

I have used three different environments for developing train control software in Ada. The Ada programs for the rtVAX target were developed using VAXELN Ada under the VMS operating system. The Ada programs for the x86 targets have been developed using both IBM Rational APEX with the Rational Exec runtime and GNAT GPL with the MaRTE [9] runtime.

10 Summary

The model railroad provides an exciting environment for teaching a real-time embedded systems course – a course in which most student teams successfully complete a major software project. I have developed the interface hardware to allow other schools to easily connect a variety of computers to a model railroad at minimal cost. Contact me for more information. In addition to circuit and wiring details, I can supply a spreadsheet that you may use to produce part lists and cost estimates for whatever size railroad you care to build. A sample user's manual, photographs, and a video of the laboratory described in this paper may be found at http://www.cs.uni.edu/~mccormic/RealTime/.

References

[1] Burns, and A. Wellings (2001), *Real-Time Systems and Programming Languages* (3rd Edition), Addison Wesley.

[2] M. Amirijoo, A. Tešanović, and S. Nadjim-Tehrani (2004) *Raising Motivation in Real-Time Laboratories: The Soccer Scenario*, Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Norfolk, Virginia, USA, pp 265-269.

[3] J. W. McCormick (1988), *Using a Model Railroad to Teach Digital Process Control*, SIGCSE Bulletin, Vol 20, pp 304-308.

[4] J. W. McCormick (1991), *A Laboratory for Teaching the Development of Real-Time Software Systems*, SIGCSE Bulletin, vol 23, pp 260-264.

[5] J. W. McCormick (2005), *We've Been Working On The Railroad: A Laboratory For Real-Time Embedded Systems*, Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, St. Louis, Missouri, USA, pp 530-534.

[6] Dallee Electronics, Inc. Railroad Sound Systems. http://www.dallee.com/sound_systems.htm

[7] M. Eisenstadt (1997), *My Hairiest Bug War Stories*, Communications of the ACM, vol 40, no 4, pp 30-37.

[8] J. W. McCormick (1997), Forum Letter, Communications of the ACM, vol 40, no 8, p 30.

[9] M. A. Rivas and M. G. Harbour (2001), *MaRTE OS: An Ada Kernel for Real-Time Embedded Applications* in G. Goos, J Hartmanis and J. van Leeuwen (eds) Reliable Software Technologies — Ada-Europe 2001 in LNCS vol 2043, pp 305-316, Springer-Verlag.

Is Ada Education Important?

Jean-Pierre Rosen
Adalog, 19-21 rue du 8 mai 1945, 94110 Arcueil, France; email: rosen@adalog.fr

Abstract

Some things seem to go without saying. Who would argue that education, on any technique, is not important? However, there can be many factors beyond education that can affect the success of a computer project, or the popularity of a programming language. In this paper, we are assessing the importance of education, and especially Ada education, compared to other factors, for the achievement of the goals of Ada.

Introduction

As a consultant, I meet lots of people who are busy developing software in Ada. Few of them know more than the very basics of the language – and typically what makes Ada different from other programming languages.

When asked about the kind of Ada education they received, quite often the answer is: "none". Let's face it: companies do not see training as an investment, but as a cost. Very often, when a resource is needed on an Ada project, the person is given a book (at best Barnes', at worst a small book about preliminary Ada, published circa 1980[7]) and told to learn the language by himself.

On one hand, one can consider that an experienced programmer should be able to learn any language. But this requires, at least, an "experienced" programmer. And on the other hand, Ada was specifically designed to change the way software is written. While it is easy to learn the syntax of the language, learning how to use it efficiently, in a way that serves its design purposes, is a quite different issue.

At this time, where the community is still wondering why it is so hard to get Ada accepted, where we see the language war still raging, where many new languages still lack basic features that were already in Ada 24 years ago, where we see very high-level "model driven" designs that are still based on bytes and words as soon as something can be expressed as an integer value, it seems important to explore the role of education in the success of Ada (or lack of thereof), and in software engineering in general.

Thesis: Ada education is important

As mentioned above, many people had few, if any, Ada education before they were assigned to an Ada project. Those who join an existing project are influenced by the existing structure, they can meet more senior developers with better language experience, and thus acquire some basic skills. But quite often, those people are directly in charge of full developments, ranging from a single module to a whole application.

Consequence of poor education: bad language usage

People naturally try to import directly the programming style from languages they are familiar with; and of course, they do not use features that they never heard about. Things like aggregates, array slices, or simply full array assignments are not used. An old proverb says:

> *A good FORTRAN programmer can write FORTRAN in any language.*

Most people know only two basic data types: Integer and Boolean. Actually, convincing people that Integer is just a normal integer type which is not especially interesting is one of the most difficult messages to teach; there seem to be an implicit belief that Integer is more or less the mathematical notion of integer number. And Boolean is for all cases that need to memorize some form of state. In one of the exercises that are part of my regular Ada training session, we come across a small state machine with three states. Each time, about half of the students represent the states as a set of three booleans. Even the notion of an enumerated type does not seem natural to them.

Some symptoms are characteristic of a C background [8]; I once met a program where all string manipulations were done by passing the address of the first character to the routine, then Uncheck_Converting it to pointer-to-characters to access the characters, or to Integer to increment it… Another funny example is a program where the bit (used as a constant for representation clauses) was defined as 1/8th of Character'Size – clearly because, in C, all sizes are defined as multiples of the size of char.

In another project, people clearly had a hard time understanding strong typing. To be honest, they did not understand it at all: each time the compiler complained that types did not match, they simply used Unchecked_Conversion to shut up the compiler. Eventually, using Unchecked_Conversion became such an habit that it was used even between different subtypes of the same type…

But the most contrived example I came across was the following. Imagine you have a string of one character, and

[7] No kidding, I've seen that recently.

[8] All the following examples are taken from actual projects – many of them being DO178B level A or B

you want to assign a character into it. How would you do? Here is the answer:

```
  subtype String_One is String (1..1);
  V : String_One;
  C : Character;
  function C_to_S is new
    Unchecked_Conversion (Character, String_One);
begin
  V (1..1) := C_To_S (C);
```

Consequence of bad language usage: bad perception of Ada

This last example is extremely worrisome. Independently from the fact that it worked almost by miracle, it is easy to imagine that the programmer who found this solution did not get there easily; he must have gone through a lot of trials and errors, fighting furiously Ada's typing system, until he eventually found a combination that compiled and (hum) worked.

Now, what do you think that this guy told to his fellow co-workers later? That there must have been a better solution and that he needed extra training to use the language properly? Certainly not. Most likely, he told them that Ada is a horrible language, where you have to jump through unbelievable hoops to do even the simplest things, that are soooo straightforward in C.

There is also a nasty consequence of bad Ada style, regarding efficiency. In C (or Fortran), there is no array assignment. Therefore, compilers are very clever at recognizing patterns that can be optimized, like:

```
for (I = 0; I < N; I++)
  A[I] = B[I]
```

Note that from a theoretical point of view, there is a real abstraction inversion here, since the compiler recognizes a high level statement (an array assignment) from the detailed description of its implementation.

Of course, recognizing this kind of pattern is much less important in Ada, since programmers are expected to simply write:

```
A := B;
```

Therefore, compiler writers may spend their precious time into optimizing other useful constructs rather than loops that assign whole arrays element by element. But if a programmer compares the relative speeds of the same program written in C and Ada, with an almost word-to-word translation of the C into Ada (thus keeping this kind of loop), Ada will certainly be disadvantaged (not counting other effects, like comparing a checked version of the Ada code with an unchecked version of the C code). Of course, the same could be said of C, if the experience was attempted the other way round: if you try to hand-code into a C program the same level of security that's automatically provided by Ada, there would be a huge time penalty for C; but in practice, it never happens this way. People just write the C program, "translate" it (badly) into Ada, measure, and make the general conclusion: "Ada is slower".

Not only do these Ada-illiterate (or ill-literate?) people spread bad words; they prevent their projects from achieving the promised benefits from Ada. Imagine a quite likely story. A project manager hears about Ada, the safety that it brings, the gain in development and debugging costs, etc. Being not specially opposed to novelty, he decides to try Ada for his next project – but not a big one, to minimize the risks. Since there is not much money in the project, and he has a team of nice and experienced C programmers, he just buys a book about Ada and gives it to the team. The programmers, in turn, don't have much time to study the book in depth, so they just wander trough the syntax, write one or two "hello world" type programs, and think they can go on with the project.

Of course, their programming style will just be "C in Ada". No real use of strong typing, poor decomposition into packages, lots of hacks with addresses, unnecessary loops, etc. resulting in clumsy coding, difficulties to get the code through the compiler, and poor performance. Will the resulting program be safer? Unlikely. Will it be more readable than it's C equivalent? Presumably not. Will there be any saving in development and debugging costs? No. What will the project manager conclude? "Ada is just another one of these hyped technologies that does not work in practice – let's return to C".

This is something that the Ada community must recognize: if Ada is not more widespread, it is because people did *not* get the touted benefits. Without education, the power of Ada cannot be realized, and if not realized, there is no power in Ada! As noted by Grady Booch [1]:

> *Give a power drill to a carpenter who knows nothing about electricity, and he would use it as a hammer. He will end up bending quite a few nails and smashing several fingers, for a power drill makes a lousy hammer*

Antithesis: Ada education is not important

Does it make sense to teach how to use a power drill to people who have not access to electricity?

Teaching CS is the first (missing) step

Ada is not a goal in itself, it is only a tool intended to serve design methodologies. If people are trained into software engineering, understanding the principles of Ada is easy. Teaching Ada to QA people, for example, is a breeze. Each time you explain the motivation and reasons for a particular feature, those people nod with sympathy, as you are addressing exactly their day-to-day concerns.

But, as noted by Dewar and Schonberg in a recent paper [3], there is a general lack of education in CS in general. Is there any sense in teaching Ada to people who have no idea about the goals it is intended to serve?

There is, in the new generation of programmers, a considerable lack of education in algorithmics in general, even for the most basic knowledge expected from anyone who graduates in CS. In every training session, I have at least half of the attendees who are unable to do simple pointer manipulations, like putting an element in front of a simple linked list. Saying that something should be avoided because it would go $O(n^2)$ is like talking a foreign language. And most of them never heard about the dining philosophers…

But it is not only the basic skills that are missing. After all, these are just technical stuff, and these people should be able (should!), when needed, to find the necessary algorithms in the literature – assuming they have enough notions of algorithmics to *think* about looking for an existing solution. More of a concern is the fact that most developers have no consciousness of the stakes of software and of the consequences of what they are doing.

For example, a colleague of mine was in charge of integrating some code. He came across a situation where he found a subprogram that had to be executed with mutual exclusion. To that effect, the subprogram was passed a pointer to a semaphore in its parameters. In some occasions, there was a bug on the caller side, and the subprogram was passed a null pointer, resulting in a crash (it was C++). Since my colleague was in charge of finding the problems, but not fixing them, he made a detailed report of what was happening to the developers team. Sometimes later, he received the "fixed" version: in the subprogram, a test was made, and the sequence that grabbed the semaphore bypassed if the pointer was null.

The really frightening moral of this story is that the person who "fixed" the code simply thought: "we have a crash because we dereference a null pointer, therefore let's skip the sequence if the pointer is null". It didn't cross anybody's mind to ask: "well, if there is a semaphore here, it must be for some purpose". People are fixing the symptoms, but there is a total lack of *reasoning* about the logic of what they are doing. As R. Dewar likes to say:

> *When Roman engineers built a bridge, they had to stand under it while the first legion marched across. If programmers today worked under similar ground rules, they might well find themselves getting much more interested in Ada!*

The role of methods

Teaching syntax is easy. Teaching coding is a bit more difficult, but is achievable with any programming language and any audience. But how do you train people into *thinking right*?

For a long time, it was considered the job of design methods. A design method was intended to provide a formal framework to drive and guide the design process. A typical example of this was the HOOD design methods; in its early versions, it was organized as a succession of steps, each defining what should be performed (defining the properties of objects, their relations to other objects, etc.), requiring precise documentation, the establishment of traceability documents, etc. But the method was not easily accepted, and over time, became less and less constraining. In its latest form, the method just "suggests" an "example of design process". What happened? Programmers did not like rigid frameworks that "hampered creativity". The dialog between Programmer and Method ended up like this:

Method:	Thou shallst not do it this way.
Programmer:	But I want to do it like that!
Method:	No, I am the Method, and thou shallst obey by my commandments
Programmer:	OK then, I'll pick up another method.

Nowadays, methods have moved away from guiding reasoning, and focus on providing tools and notations. UML (which can hardly be called a method – actually the "L" in its name means it is only a language) is intended to provide all diagrams a programmer could ever need to represent any design. There is not a word about *how* it should be used; it is simply not the purpose of UML. Maybe that's why UML is so popular: anybody can do UML without changing his (bad) habits.

Tools and notations are important helpers to *support* reasoning – but they do not *replace* reasoning. Proof making tools are important to ensure a design is right – but they are of no help to make the initial design. MDA allows model transformation from a high level model down to a machine-executable one – but still, someone has to design the initial model, and the initial model has to be correct. Whatever the tool, the old saying applies:

> *Garbage in => garbage out*

From this point of view, Ada is just another tool to support software engineering. As any methodological tool, it was designed to *avoid* inconsistencies, *diagnose* bad designs, and *prevent* people from using the (bad) habits of other languages. Enjoying all these benefits (or simply considering them as beneficial) assumes that people understand the processes of design; it is not surprising that uneducated people experience a lot of difficulties with the language. Is it possible to help these people by explaining the difference between a primitive and a non-primitive operation? Certainly not. Can we attract programmers by touting all the nice *technical* features of the language? No way, as long as the prospective users don't have the necessary knowledge to match features against the purpose they serve.

Ada education is not important for the success of Ada; educating people in software engineering, promoting rigorous design and correctness by construction, teaching algorithmics (in the sense of teaching people how to take an algorithmic approach to problem solving), etc. are the keys to the motivation that will make Ada not only acceptable, but the language of choice for the *implementation* of these concepts.

Why do people focus on the language after all?

Finally, it should be stressed that insisting on Ada education can have a negative effect. Many project managers would ask: "why should we choose a language that requires special education, when we can choose a language that any beginner can use?" Or even worse: "if only experts can use Ada, we'd better stay away from it".

We all met project leaders who are turning away from Ada because they cannot find enough trained programmers[9]. The strange thing is that if you ask them if it is easy to find resources who are educated in designing critical systems, in real time, in system programming, and the like, they will just not consider these skills as important on a resume. It is however much more difficult to train someone into *thinking secure* than to teach a programming language – even Ada. Why do recruiters focus only on programming languages is one of the great mysteries of computer science.

Synthesis: Good news and bad news

Good news: there is no problem with Ada

Every now and then, on comp.lang.ada or other places, we see claims by various people that "if only Ada had such and such wonderful feature, then it would be immediately successful". I've never met someone in the industry who told me that the reason for not using Ada was missing functionalities in the language. Although there might be some rough edges here or there, the overall language design is good. And it serves its purpose well: making a bad design almost impossible to compile.

Actually, a majority of Ada programmers are not familiar with *all* the possibilities of the language. As a consultant, it is my duty to help people by telling them *how* to use the language, and especially by making them aware of programming patterns and language tools that they didn't even consider. Every time I was given a problem, I found the necessary resources in the language to solve it in a satisfactory way. There is really no need for running after more and more new functionalities.

Bad news: the problem is much bigger than the language

Who are today's programmers? We can distinguish roughly the following categories:

- Old-timers, who have learned to squeeze every possible micro-second out of assembly language, are perfectly happy with a command line and vi, and who have a hard time with object orientation, GUIs, and many modern things that *are* necessary in modern developments.
- There is still a number of people coming from various backgrounds (electronics, chemistry…) who became programmers because there were more jobs there than in their branch, without any education at all.
- There is the new generation, who learned programming with Java and think that programming is just connecting a button to a "start" method of a component.
- And a small number of people who still believe in software engineering and who are wondering how to make sense out of this mess: you and me…

In the old time, people wrote obfuscated assembly code, and were proud of putting a comment saying "this code is impossible to understand, but don't worry, it works". Now, with UML, MDA, eXtreme Programming, and the like, the same people are proud of making obfuscated class diagrams. I've seen projects where the walls were covered with class diagrams, and I found this frightening. As Booch stated [2],

> *The task of the software development team is to engineer the illusion of simplicity.*

Making things simple is difficult, and the mess is the same as 30 years ago! In 1972 [4], Dijkstra noted:

> *As long as ther were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.*

This statement equally applies nowadays – although gigantic computers of the time had much less power than the average telephone of today. The software crisis is still here.

Prosthesis: What can we do?

The ideal cursus

Ideally, bringing students to the right state of mind would need the following, in this order (OK, it's a dream):

- Teach people how to think
- Teach goals and principles of software engineering
- Teach problem solving in general
- Teach algorithmics
- In your spare time, teach Ada. At this point, it should be easy.

But of course, if you take this program seriously, you will be flooded by students asking: "sorry Professor, but it seems you miss something important in your curriculum. Where do we learn how to make nice Web pages?"

Teach programming as an engineering discipline

True enough, Ada is not "fun". It is not even easy to learn for beginners. I once read a paper that criticized Ada as not appropriate for teaching programming in elementary schools – granted.

Ada is an industrial language. The benefits of the language are in increased reliability, lower development costs, easier maintenance, etc., not in playing with funny animated gadgets. When students complain that Ada is not fun, ask them if civil engineering is fun. Do they think that engineers who build towers just design them "for pleasure", "as they feel", without any consideration for the rules of the art, and then take bids on whether the tower will collapse or

[9] While at the same time, students do not take the Ada course on the grounds that there are no jobs!

not? And if the tower falls apart, do they say "oh, there was a bug, let's start over again"?

This does not mean that there is no pleasure in programming in Ada; but Ada developers are proud of having well designed programs, that work without, or very small, debugging effort. In a sense, Ada replaces the pleasure of the game with the pleasure of a work well done.

This is far from obvious to beginners. It needs explanations, backed-up by experience, to bring one's mind to the point where Ada is really enjoyable. Simply stated, this needs understanding that software development is an engineering discipline – not a video game. Ada is for craftspersons. Maybe, the small minority of people who understand the issues of software should join and make a Union of Craftsprogrammers.

Teach Ada to those already involved in Ada projects

Ada education is important in the short term. Once a project has decided to use Ada, the best thing we can do to promote the language is to make sure that the project is successful. And the first thing to do to that effect is to make sure the language is used appropriately and effectively.

Project managers should be aware that Ada training is an investment. Without costs, there are no benefits. Training costs are small compared to the mere cost of hiring someone, and even smaller compared to the cost of a late project. But it is an upfront cost, and education is needed to accept this cost even before the project starts.

Developers should be aware of their limits, and seek advice of more experienced people or external consultants when they hit some difficulty. Ada is a vast language, it provides all the necessary tools, the difficulty is to find the right tool to do the job – something that requires experience.

Promote software engineering to others

In the long term, the problem of education goes far beyond Ada only. As should be clear from the above discussion, it makes no sense to promote Ada until people are aware of its benefits – which means they have a clear understanding of the goals and methods of software engineering. Selling Ada alone can even hurt the language, because Ada alone cannot solve all the problems, and people who try the language without success will actively spread bad words about Ada – it is easier to blame failure on the language than on one's own inability to master the course of the project.

If you succeed in promoting the "good spirit", you can introduce, as an engineering decision, the choice of the tool (i.e. the language) used by the project. Ask the project leader to make an impartial spreadsheet weighing the pros and cons of several languages. Don't expect Ada to win all the times – if Ada wins often enough, it's enough!

References

[1] Grady Booch, *Object Oriented Design with applications*, p. 33

[2] Grady Booch, *Object Oriented Design with applications*, p. 23

[3] R. Dewar and E. Schonberg, *Computer Science Education: Where Are the Software Engineers of Tomorrow?*, CrossTalk, Jan. 2008

[4] E. Dijkstra, Turing Award Lecture

Use of Ada in a Student CubeSat Project

Carl Brandon

Vermont Technical College, PO Box 500, Randolph Center, VT 05061 USA; Tel: +1 802 728 1350; email: carl.brandon@vtc.edu

Abstract

A student project to develop a CubeSat (10 cm cube, 1 kilogram satellite) as part of Vermont Technical College's Aeronautical Engineering Technology degree uses a Texas Instrument MSP430 processor for which no Ada compiler is available. Ada and SPARK offer a highly desirable combination for the reliability needed in a satellite. Since there is no Ada compiler targeted for the MSP 430, we decided to use SofCheck's AdaMagic compiler which generates ANSI C as its intermediate language. We then use an existing C compiler as a back end, generating code for the MSP 430. This allows the students to write the original source code in Ada/SPARK and have object code for the MSP430.

Keywords: CubeSat, SPARK, Ada.

1 CubeSats

A CubeSat is a pico satellite approximately the size of a 10-centimeter cube, with a maximum mass of one kilogram. The particular specification for the satellite hardware was developed by California Polytechnic State University (Cal Poly) and Stanford University (http://CubeSat.calpoly.edu), so that multiple CubeSats could be easily integrated into a launch vehicle.

The CubeSat is an autonomous satellite. The software to run all its systems must be completely reliable. It will be powered by high-efficiency triple junction photovoltaic cells, backed up by batteries for high-power operations, such as transmitting and, if the satellite is behind the earth, out of the sunlight power supply.

The Vermont Tech CubeSat will have a 2.4 GHz Microhard (http://microhardcorp.com/MHX2420.htm) spread spectrum modem for two-way communications with our ground station. This radio has been used in two CubeSats so far. Although the radio has a lot of built-in autonomous functionality, it will be controlled by the flight module computer.

Another function, the attitude determination and control system, will have the largest software component of any onboard system. We plan to have an active magnetic attitude control system. This will be used to point the patch antenna for the transceiver, and the camera toward the earth.

The position of the Cubesat after deployment is generally determined by calculation from the Keplerian two-line elements describing the orbit as released by the launch providers soon after deployment. A program, such as Satellite Tool Kit (http://www.agi.com/index.cfm) can give accurate position, assuming the two-line elements are correct. The position accuracy is important for antenna pointing of the ground station to establish communications with the satellite. We will achieve even higher accuracy by including a Global Positioning Satellite navigation module in the CubeSat to transmit a more precise location of the satellite, which will correct the orbital parameters of the Keplerian two line elements.

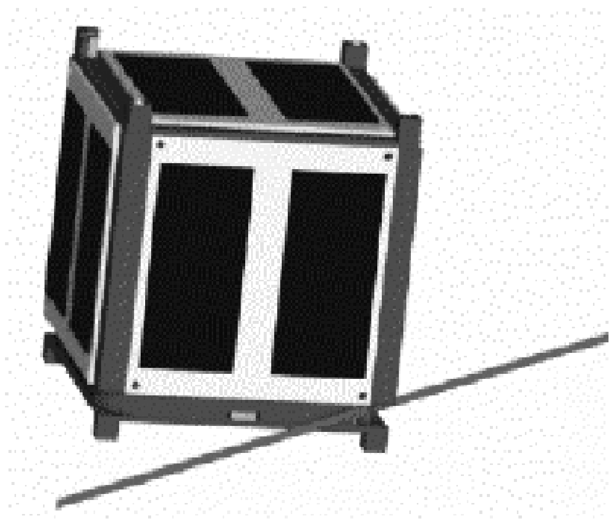

Figure 1 A CubeSat

The CPU for the satellite will be a Texas Instruments MSP430 micro controller, chosen for its extremely low power consumption, the lowest of any processor. With power production from the photovoltaics being about 1.5 watts in the sunlight, low power is an absolute requirement. The primary disadvantage, in our view, is that there is no Ada compiler for this processor. As discussed below, Ada/SPARK is our choice for the satellite software.

Cal Poly has developed a deployment system called a P-Pod, which holds three CubeSats, and releases them via a spring from the launch vehicle at the appropriate time. Cal Poly negotiates with commercial launch providers for P-Pod space on commercial satellite launches (there has also been a CubeSat deployment from the Space Shuttle). They negotiate launch prices in the $30,000-$50,000 range for a one kilogram CubeSat.

2 Arctic Sea Ice Buoys

We have been funded for a cooperative NASA grant with the University of Vermont, for a prototype Arctic Sea Ice Buoy which will use the same CPU and some of the software used in the Cubesat. We have also submitted a second cooperative NASA grant application with the University of Vermont, which will fund the construction and deployment of ten Arctic Sea Ice Buoys. These would be placed on ice in the Arctic Ocean to monitor wind speed and direction, temperature and GPS position of the buoy, and relay the data via the Iridium satellite network. The buoys share some of the same characteristics of the CubeSat: low power availability, harsh environmental conditions and the need to be reliable and autonomous.

Software for control of the radio, power management, and telemetry will be shared from the CubeSat software development. The same necessity for extreme software reliability speaks for the use of Ada/SPARK.

3 Aeronautical Engineering Technology Degree Program

The general design of the CubeSat is being done by students in our Aeronautical Engineering Technology program. They take two semesters of Spacecraft Technology and a satellite design lab. In the lab, they look at the various satellite systems: command and control (CPU), attitude determination and control, communications, power and instrumentation. These associate degree students generate general specifications for the satellite, but the implementation will be done by bachelor students in our Electro-Mechanical, Computer Tech and Software Engineering programs during their senior projects.

4 Our CubeSat Project

With the somewhat limited personnel resources of a small college (Vermont Tech has about 1,500 students), we have chosen to use as much off-the-shelf technology in our spacecraft as possible. We have started with a CubeSat kit (http://www.CubeSatkit.com) which supplies the hardware chassis (flight module), the CPU board, real time operating system (Salvo), a number of software components, and a development board.

We are purchasing our electrical power system, which provides batteries, charging controller, and telemetry data from Clyde Space (http://www.clyde-space.com). The photovoltaic cells are TASC cells from Spectrolab (http://www.spectrolab.com). We will be fabricating our own PC boards for mounting the cells. These boards will also make up the outer shell of the satellite.

The spacecraft's attitude will be determined by a three-axis magnetometer to measure the direction of the earth's magnetic field, and a sun sensor to determine the direction of the sun. These two pieces of information, with a lot of computation, will enable the satellite to determine where it is pointing. To change its orientation, three mutually perpendicular torque coils will lie under the faces of the satellite, and computer controlled currents can be sent to any of them to create a torque against the Earth's magnetic field, and thus rotate the satellite to the desired orientation. Although there is an off-the-shelf attitude determination and control system similar to what we want, it is much more complex (containing three torque wheels), adds a second ten-centimeter module, increasing the launch costs by $30,000-$50,000, and costs $55,000 itself.

For the communication system, we will use the Microhard 2.4GHZ spread spectrum modem, and possibly a second radio beacon in the 440 MHz amateur radio band. Students will build a tracking dish antenna of three to eight meters, and a tracking dual yagi antenna for 440 MHz. Our ground station will become part of the GENSO (http://genso.org) network when it becomes operational in the fall of 2008.

The final instrument payload is yet to be determined, but will most likely include a camera for photographing the earth from space. Other instruments may be included, and all will have to be controlled, and data collected by the CPU.

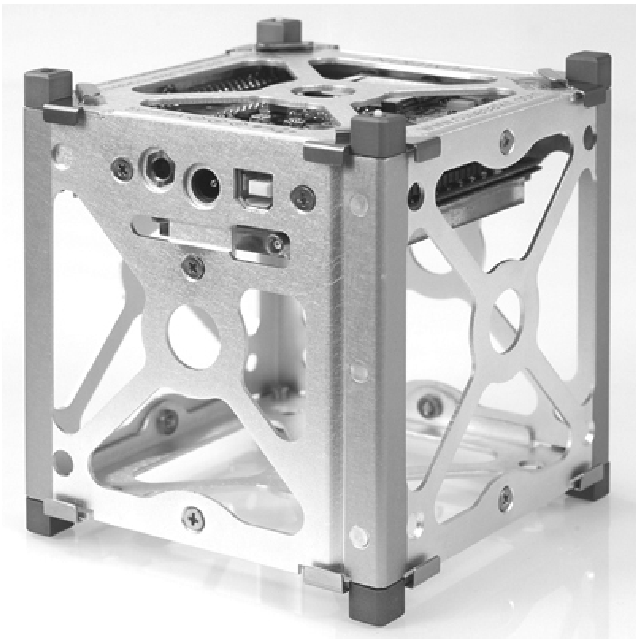
Figure 2 CubeSat kit

5 MSP430 CPU Description

The Texas Instruments (http://focus.ti.com) MSP430 series of micro controllers are the lowest power micro controller/processors available. This makes them an excellent choice in an application where power is limited. They also contain a variety of peripherals on the chip, which also saves on complexity and power. The peripherals vary with the specific chip, but we are looking at a final choice of the MSP430F2618 which has 116 kb of flash, 8 kb of RAM, 12 bit SAR analog to digital converter, 2 12 bit digital to analog converters, analog comparator, DMA, Hardware Multiplier, 2 USCI interfaces. The CPU board can also take SD flash memory cards of up to 2 GB.

It uses 2 µA in low-power mode, and about 500 µA at full speed. It can go from low-power mode to full speed in one microsecond. There is no Ada compiler for the MSP430.

6 Why use Ada

Although the CubeSat is not a safety-critical system, the software is mission critical. The small size of the CubeSat precludes uploading software patches as is sometimes done with NASA satellites and space probes. The cost of developing the CubeSat will be in the $30,000-$50,000 range, and the launch costs also in the $30,000-$50,000 range, so a non functional satellite because of a software error would result in a $60,000-$100,000 loss. Most of the CubeSats launched to date have been programmed in C, and admittedly most have generally worked.

Despite the general success of CubeSats programmed in C, Ada offers a number of advantages. Many large projects programmed in Ada have shown considerable reductions in error rates compared to C. In addition, finding and fixing the errors that do occur takes much less time. With the small size of our school, and thus fewer people resources for the project, efficiencies of this type are very important. In addition, having students involved in both the hardware and software for the project and using a language that makes use of the best of software engineering features has a great pedagogical advantage. This project is a real-embedded system that must have very high-integrity software. Ada fits the bill for high-integrity software that is efficient to write and debug.

7 Ada and SPARK

The availability of SPARK makes possible a further increase in the integrity of the code over Ada alone. SPARK annotations allow the specification of the program, as expressed in the annotations, to be used by the SPARK toolset to check the code's compliance with the specification. Although this project is rather small compared to the projects that SPARK is normally used on, the fact that there is only one chance to get the deployed software right, and the high cost of failure in dollars and time, make it a good choice to help ensure the success of our project.

A second benefit of using SPARK is that this is an opportunity for some of our Software Engineering majors to work on a high-integrity real world project. This is a type of project not often done in an academic environment. SPARK allows the students to get experience with a particularly powerful method of achieving high-integrity software.

In the CubeSat, there is the need for real-time programming. There are interactions with the power system, the attitude determination and control system, the communications system, the navigation system and the camera and other instrumentation. The availability of RavenSPARK, the SPARK subset of the Ada Ravenscar Profile, will allow us to use SPARK to keep the real-time programming also very high-integrity. Thus the students will have a valuable experience in writing robust and clear software, that otherwise would not be available to them.

SPARK, being a subset of Ada, requires an Ada compiler. The only problem is that there is no Ada compiler for the processor we want to use.

8 AdaMagic

A solution to the compiler problem for this project required an unusual process. In talking with Tucker Taft of SofCheck (http://www.sofcheck.com) at Ada Europe 2005, I learned about their AdaMagic compiler which produced ANSI C code as the intermediate language. This opened the interesting possibility of using an existing ANSI C compiler as the "back end" for the AdaMagic compiler. This is the route we have chosen, so we can develop software for our CubeSat in Ada/SPARK for high-integrity, check it with the Ada and SPARK toolsets, run it through AdaMagic, and then compile the resulting ANSI C version with our C compiler for the MSP430.

9 Crossworks C Compiler and the Salvo operating system

We have chosen the Rowley Associates (http://www.rowley.co.uk) CrossWorks for MSP430, which includes an ANSI C compiler, macro assembler, linker/locator, libraries, core simulator, flash downloader, JTAG debugger, and an integrated development environment, CrossStudio. This will provide the object code for our satellite CPU and download it into the processor. There is also an MSP430 core simulator, so code can be checked on the host Windows machine before downloading to the MSP430.

CrossWorks supports the Salvo Real-Time Operating System from Pumpkin, Inc. (http://www.pumpkininc.com), the manufacturer of the CubeSat kit. Salvo comes with the CubeSat kit, and will be the operating system in the satellite. Several in orbit CubeSats are running on Salvo.

10 GPS, GNAT Pro and SPARK

The development process is greatly facilitated by the inclusion of a set of Python scripts with the GNAT Programming Studio that allow invocation of the various command line tools in the SPARK toolset from within GPS. Having first used the SPARK toolset by invoking it from the command prompt window, this facility is extremely useful. One can now remain in the GPS development environment when using the SPARK tools.

11 Python Scripts for AdaMagic in GPS

At the moment, we are invoking AdaMagic from the command prompt window, as we used to do with SPARK. In our programming languages course, the students study both Ada and Python. In the coming semester, they will be assigned a Python project to create the necessary Python scripts to invoke the AdaMagic front end from within GPS. This will allow the programmer to remain in the GNAT Programming Studio environment for the entire development process for a much nicer overall process.

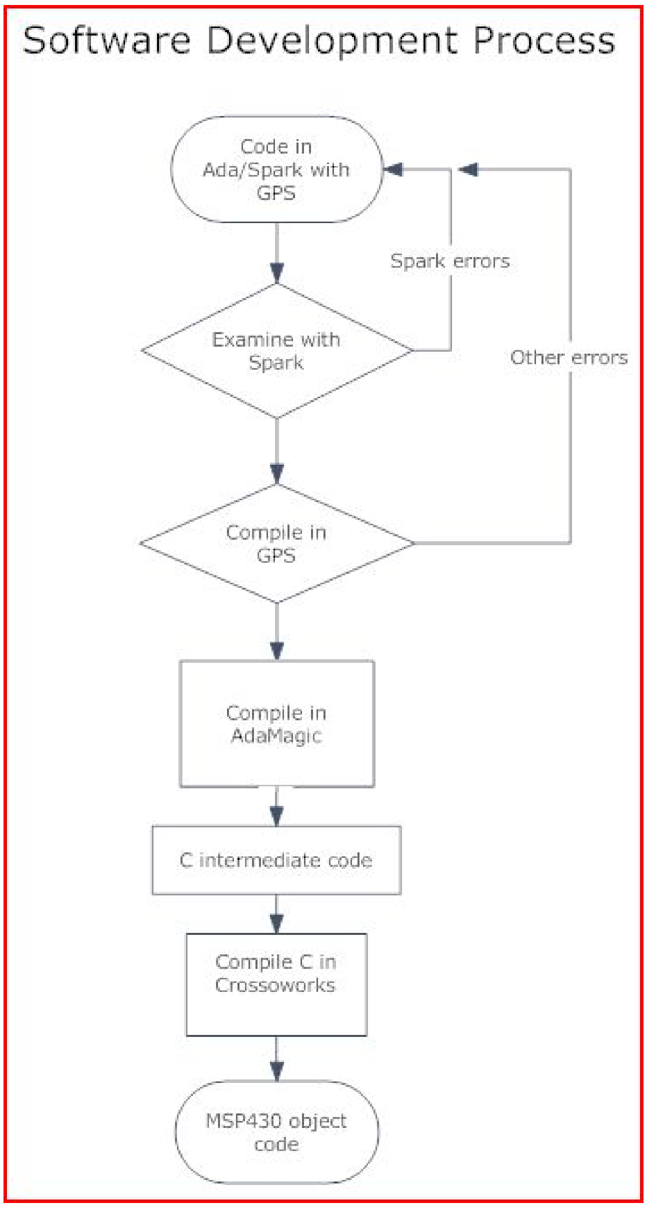

Figure 3 Software development process

12 Software workflow

Alongside is a diagram of the software development workflow with our various software tools. This shows the original development in GPS, use of the SPARK tools, compilation in GPS with GNAT, compilation again with AdaMagic with the production of ANSI C code and cross compiling with Crossworks for the production of the MSP430 object code.

13 Acknowledgments

This project at a school of our size would not be possible to undertake without a lot of outside support. For the satellite and supporting hardware, several grants have been obtained from the Vermont Space Grant Consortium, a part of NASA Space Grant (http://www.vtspacegrant.org). This would also be the source of funds for the satellite launch.

Analysis of the satellite in many aspects, from orbital analysis to thermal and communication link budget analysis, is being done in Satellite Tool Kit,. We have received a donation of thirty copies of this $100,000-per-copy software.

We have been a long time member of Adacore's GNAT Academic Program (GAP) (http://www.adacore.com/home/academia) and have had the availability of GPS/GNAT for several years. We have also received a grant of GNAT Pro from Adacore for use in our grant-funded research.

Praxis (http://www.praxis-his.com) has donated the $122,000-per-copy Spark toolset to us through our membership in GAP.

Rowley Associates has given us an academic discount for one license for Crossworks, and donated a second license.

We are very grateful to all these organizations for their extraordinarily generous gifts to Vermont Technical College. Without their support of academia, this project would have never launched.

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 #27 — Changing Data Representation (Part 1, Automatic Representation Changes)

Robert Dewar, AdaCore

Date: 3 March 2008

Let's get started…

A powerful feature of Ada is the ability to specify the exact data layout. This is particularly important when you have an external device or program that requires a very specific format. Some examples are:

```
type Com_Packet is record
   Key : Boolean;
   Id  : Character;
   Val : Integer range 100 .. 227;
end record;

for Com_Packet use record
   Key at 0 range 0 .. 0;
   Id  at 0 range 1 .. 8;
   Val at 0 range 9 .. 15;
end record;
```

which lays out the fields of a record, and in the case of Val, forces a biased representation in which all zero bits represents 100. Another example is:

```
type Val is (A,B,C,D,E,F,G,H);
type Arr is array (1 .. 16) of Val;
for Arr'Component_Size use 3;
```

which forces the components to take only 3 bits, crossing byte boundaries as needed. A final example is:

```
type Status is (Off, On, Unknown);
for Status use (Off => 2#001#, On => 2#010#,
                Unknown => 2#100#);
```

which allows specified values for an enumeration type, instead of the efficient default values of 0,1,2.

In all these cases, we might use these representation clauses to match external specifications, which can be very useful. The disadvantage of such layouts is that they are inefficient, and accessing individual components, or in the case of the enumeration type, looping through the values, can increase space and time requirements for the program code.

One approach that is often effective is to read or write the data in question in this specified form, but internally in the program represent the data in the normal default layout, allowing efficient access, and do all internal computations with this more efficient form.

To follow this approach, you will need to convert between the efficient format and the specified format. Ada provides a very convenient method for doing this, as described in RM 13.6 "Change of Representation".

The idea is to use type derivation, where one type has the specified format and the other has the normal default format. For instance for the array case above, we would write:

```
type Val is (A,B,C,D,E,F,G,H);
type Arr is array (1 .. 16) of Val;

type External_Arr is new Arr;
for External_Arr'Component_Size use 3;
```

Now we read and write the data using the External_Arr type. When we want to convert to the efficient form, Arr, we simply use a type conversion.

```
Input_Data  : External_Arr;
Work_Data   : Arr;
Output_Data : External_Arr;

(read data into Input_Data)

- -  Now convert to internal form
Work_Data := Arr (Input_Data);

(computations using efficient Work_Data form)

- -  Convert back to external form
Output_Data := External_Arr (Work_Data);
```

Using this approach, the quite complex task of copying all the data of the array from one form to another, with all the necessary masking and shift operations, is completely automatic.

Similar code can be used in the record and enumeration type cases. It is even possible to specify two different representations for the two types, and convert from one form to the other, as in:

```
type Status_In is (Off, On, Unknown);
type Status_Out is new Status_In;

for Status_In use (Off => 2#001#, On => 2#010#,
                   Unknown => 2#100#);
for Status_Out use (Off => 103, On => 1045,
                    Unknown => 7700);
```

There are two restrictions that must be kept in mind when using this feature. First, you have to use a derived type. You can't put representation clauses on subtypes, which means that the conversion must always be explicit. Second, there is a rule RM 13.1(10) that restricts the placement of interesting representation clauses:

> 10 For an untagged derived type, no type-related representation items are allowed if the parent type is a by-reference type, or has any user-defined primitive subprograms.

All the representation clauses that are interesting from the point of view of change of representation are "type related", so for example, the following sequence would be illegal:

```
type Val is (A,B,C,D,E,F,G,H);
type Arr is array (1 .. 16) of Val;

procedure Rearrange (Arg : in out Arr);

type External_Arr is new Arr;
for External_Arr'Component_Size use 3;
```

Why these restrictions? Well the answer is a little complex, and has to do with efficiency considerations, which we will address in next week's GEM.

Ada Gem #28 — Changing Data Representation (Part 2, Efficiency Considerations)

Robert Dewar, AdaCore

Date: 17 March 2008

Let's get started…

Last week, we discussed the use of derived types and representation clauses to achieve automatic change of representation. More accurately, this feature is not completely automatic, since it requires you to write an explicit conversion. In fact there is a principle behind the design here which says that a change of representation should never occur implicitly behind the back of the programmer without such an explicit request by means of a type conversion.

The reason for that is that the change of representation operation can be very expensive, since in general it can require component by component copying, changing the representation on each comoponent.

Let's have a look at the -gnatG expanded code to see what is hidden under the covers here. For example, the conversion Arr (Input_Data) from last week's example generates the following expanded code:

```
B26b : declare
  [subtype p__TarrD1 is integer range 1 .. 16]
  R25b : p__TarrD1 := 1;
begin
  for L24b in 1 .. 16 loop
    [subtype p__arr___XP3 is
      system__unsigned_types__long_long_unsigned
            range 0 .. 16#FFFF_FFFF_FFFF#]
    work_data := p__arr___XP3!((work_data and not
                 shift_left!( 16#7#, 3 * (integer(L24b –
                 1)))) or shift_left!(p__arr___XP3!
                 (input_data (R25b)), 3 * (integer(L24b –
                 1))));
    R25b := p__TarrD1'succ(R25b);
  end loop;
end B26b;
```

That's pretty horrible! In fact one of the Ada experts here thought that it was too gruesome and suggested simplifying it for this gem, but we have left it in its original form, so that you can see why it is nice to let the compiler generate all this stuff so you don't have to worry about it yourself.

Given that the conversion can be pretty inefficient, you don't want to convert backwards and forwards more than you have to, and the whole approach is only worth while if will be doing extensive computations involving the value.

The expense of the conversion explains two aspects of this feature that are not obvious. First, why do we require derived types instead of just allowing subtypes to have different representations, avoiding the need for an explicit conversion?

The answer is precisely that the conversions are expensive, and you don't want them happening behind your back. So if you write the explicit conversion, you get all the gobbledygook listed above, but you can be sure that this never happens unless you explicitly ask for it.

This also explains the restriction we mentioned in last week's gem from RM 13.1(10):

> 10 For an untagged derived type, no type-related representation items are allowed if the parent type is a by-reference type, or has any user-defined primitive subprograms.

It turns out this restriction is all about avoiding implicit changes of representation. Let's have a look at how type derivation works when there are primitive subprograms defined at the point of derivation. Connsider this example:

```
type My_Int_1 is range 1 .. 10;

function Odd (Arg : My_Int_1) return Boolean;

type My_Int_2 is new My_Int_1;
```

Now when we do the type derivation, we inherit the function Odd for My_Int_2. But where does this function come from? We haven't written it explicitly, so the compiler somehow materializes this new implicit function. How does it do that?

We might think that a complete new function is created including a body in which My_Int_2 replaces My_Int_1, but that would be impractical and expensive. The actual mechanism avoids the need to do this by use of implicit type conversions. Suppose after the above declarations, we write:

```
Var : My_Int_2;
...
if Odd (Var) then
  …
```

The compiler translates this as:

```
Var : My_Int_2;
...
if Odd (My_Int_1 (Var)) then
  …
```

This implicit conversion is a nice trick, it means that we can get the effect of inheriting a new operation without actually having to create it. Furthermore, in a case like this, the type conversion generates no code, since My_Int_1 and My_Int_2 have the same representation.

But the whole point is that they might not have the same representation if one of them had a rep clause that made the representations different, and in this case the implicit conversion inserted by the compiler could be expensive, perhaps generating the junk we quoted above for the Arr case. Since we never want that to happen implicitly, there is a rule to prevent it.

The business of forbidding by-reference types (which includes all tagged types) is also driven by this consideration. If the representations are the same, it is fine to pass by reference, even in the presence of the conversion, but if there was a change of representation, it would force a copy, which would violate the by-reference requirement.

So to summarize these two gems, on the one hand Ada gives you a very convenient way to trigger these complex conversions between different representations. On the other hand, Ada guarantees that you never get these potentially expensive conversions happening unless you explicitly ask for them.

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o K.U. Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Peter Dencker
Steinäckerstr. 25
D-76275 Ettlingen-Spessartt
Germany
Email: dencker@web.de
URL: ada-deutschland.de

Ada-France

Association Ada-France
c/o Jérôme Hugues
Département Informatique et Réseau
École Nationale Supérieure des Télécomunications
46, rue Barrault
75634 Paris Cedex 135
France
Email: bureau@ada-france.org
URL: www.ada-france.org

Ada-Spain

attn. José Javier Gutiérrez
Ada-Spain
P.O.Box 50.403
28080-Madrid
Spain
Phone: +34-942-201-394
Fax: +34-942-201-402
Email: gutierjj@unican.es
URL: www.adaspain.org

Ada in Sweden

attn. Rei Stråhle
Saab Systems
S:t Olofsgatan 9A
SE-753 21 Uppsala
Sweden
Phone: +46 73 437 7124
Fax: +46 85 808 7260
Email: Rei.Strahle@saabgroup.com
URL: www.ada-i-sverige.se

Ada in Switzerland

attn. Ahlan Marriott
White Elephant GmbH
Postfach 327
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: ada@white-elephant.ch
URL: www.ada-switzerland.ch