ADA USER JOURNAL

Volume 25
Number 3
September 2004

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Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.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

The September 2004 issue of the Ada User Journal, which you have in your hands, is rich with valuable contents. The Articles section continues to keep a watchful eye on the progress of the Ada Language Revision process: Jim Moore gives us a fresh update on it following the latest meeting of the WG9, held in June 2004 in conjunction with the Ada-Europe Conference. The very same conference is the source of two further articles that give us further coverage of the Tutorial programme. This time we are delighted to host a report on the Ada Web Server, authored by Jean-Pierre Rosen and an outline of the Charles Container Library, offered by its very author, Matt Heaney. (The informed reader will know that the Ada Language Revision process has undertaken to include a container library in the language standard, to which decision, the existence of Charles was considerably instrumental.) The remainder of this issue contains the usual wealth of News and Events sections. I am indebted to Santiago Urueña and Dirk Craeynest for their precious service to the community in gathering the relevant information contents.

Before closing this editorial, I have to acknowledge a typographical error that occurred in the past issue of the Journal (AUJ 25-2), in the article entitled “Practical Experiences of Safety- and Security-Critical Technologies”, by Peter Amey and Adrian Hilton. The first bullet item in section 3.6 on page 100 should read: “completely unambigous” instead of the obviously erroneous “completely ambiguous”. Thanks for Garth Glynn for spotting this out. It’s heartening to know there are thorough readers out there!

Tullio Vardanega
Padova
September 2004
Email: tullio.vardanega@math.unipd.it

News

Santiago Urueña

Technical University of Madrid. Email: suruena@datsi.fi.upm.es

Contents

Ada-related Events

[The announcements reported below are a selection of the many Ada-related events organized by local groups. If you are organizing such an event, feel free to inform us as soon as possible. If you attended one please consider writing a short report for the Journal. -- su]

May 27 - Ada and Modeling + Ada-France General Assembly

From: Laurent Pautet <pautet@dorine.enst.fr>
Date: Wed, 26 May 2004 00:06:45 +0200
Subject: Thematic Day and General Assembly of Ada-France
Newsgroups: fr.comp.lang.ada

[Complete thread translated from French.]

You are kindly invited to participate in the Thematic Day « Ada and Modeling ». The schedule of the day follows later in this message. The event will conclude with the General Assembly of Ada-France, which will elect the new Board. Participation to both events is open and free (*libre*).

The event will take place at Jussieu, room 203 (building 41) on May 27, 2004. The site map is available at:

http://www.upmc.fr/FR/info/Venir_UPMC/05.

[…]

Agenda of the General Assembly:

- General matters

- Financial matters

- Election of the new Board.

- Any other business.

Some members of the current Board, notably the President, do not intend to stand for election. Consequently, potential candidates are invited to make themselves known.

Agenda of the Thematic Day

We will enjoy 9 presentations of 30 minutes each, followed by 15 minutes for Q&A. The presentations will primarily focus on applied modeling concepts and, subsequently, on the relevant technology offered to the Ada community.

09:30-10:15 D. Minguillon, CNES (presented by A. Canals CS): EAST or the libre software for Enhanced Ada SubseT

10:15-11:00 J. Desquilbet, IBM: Positioning of Ada with respect to UML and MDA

11:00-11:45 P. Dissaux, TNI-Europe: Modeling and Architectural Description with AADL

11:45-12:30 F. Normend, CNES (presented by A. Canals CS): HELIOS: experience report from the use of Ada 95

Lunch Break

13:30-14:15 M. Richard-Foy, Aonix: Modeling with Ravenscar-based design patterns

14:15-15:00 JF. Peyre, CNAM: Qasar or modeling concurrent applications with Petri Nets.

15:00-15:45 B. Maudry, Top GraphX: Modeling of distributed applications on CORBA

15:45-16:30 J. Hugues, ENST: Modeling of a CORBA ORB using Petri Nets

16:30-17:15 H. Bonnin, CS: Issues with the extraction of semantic properties with ASIS

17:15-18:30, General Assembly of Ada-France

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

From: Laurent Pautet <pautet@dorine.enst.fr>
Date: Sat, 05 Jun 2004 08:32:41 +0200
Subject: Thematic Day and General Assembly of Ada-France
Newsgroups: fr.comp.lang.ada

As agreed, the presentations have been made published on the Ada-France Web site, at:

http://www.ada-france.org/article109.html

From: Laurent Pautet <pautet@inf.enst.fr>
To: ada-france@ada-france.org
Date: Thu, 3 Jun 2004 21:00:36 +0200
Subject: Minutes of the General Assembly

General Assembly, held on May 27, 2004

Members present: L. Draghi, S. Tardieu, L. Pautet, A. Canals, E. Forterre, J-P. Rosen, P. Debondeli, G. Foliard.

Members with proxies: T. Quinot (S. Tardieu), F. Kordon (L. Draghi).

Non-members present: J. Hugues, F.Gasperoni, S. Nurdin, H. Bonnin.

L. Draghi opens the meeting recalling the events that occurred over the period 2002-3. Those events sum up in the participation in Ada-Europe 2003 and in the organisation of the Thematic Day that preceded the General Assembly. He also recalls that the objective of setting up a Web site for the Association has been achieved with success. Overall, he regrets that the activity over the last two years has been rather feeble.

S. Tardieu presents the budget results, the essential details of which can be found at:

http://www.ada-france.org/article108.html.

In particular, he recalls that, owing to the good financial health of the Association, the members in 2002 were offered free-of-charge renewal (including to Ada-Europe). The budget is approved unanimously.

He also answers a question asked by F. Gasperoni, who expressed concern for the Association not insisting on obtaining his membership. F. Gasperoni pointed out his determination to aid all associations that are related to Ada.

S. Tardieu then initiates the discussion on the formation of the new Board. L. Draghi does not wish to stand for another term in Ada-France. Several members contribute their views on the goals that the Association should have.

F. Gasperoni expresses the wish that one-day events be organised in the same way as Ada-UK (now dissolved) did in the past, or like the workshops organised in Brest by Y. Kermarrec and P. Dissaux. J-P. Rosen proposes to make stronger proselytism around Ada.

Overall, the participants voice the common will to aid the community by offering a one-day event including technical presentations and technology demonstrations, as well as the commitment to visibly operating outside the community in the context of conferences and meetings (e.g. the world meetings of the libre software).

L. Pautet points out that it may not be advisable to solely focus on libre software, at the risk of excluding industrial actors that are part of the Ada community.

The participants agree that, should no other concrete actions be promoted by the concerned industrial actors, then the Association should play the libre software 'card'.

S. Tardieu proposes that, the objectives of the Association having been clearly set out, the participants should proceed with the election of the Board. He also recalls that the founding members F. Kordon, L. Pautet and S. Tardieu are by-right members of the Board, with the goal of ensuring its future. L. Draghi notes that, in the absence of other candidatures, it will be necessary to consider the dissolution of the Association. A. Canals and E. Forterre inform that they stand up for election. S. Tardieu communicates that T. Quinot (who had to leave the meeting earlier) expressed in writing his candidature.

The resulting Board composition is approved unanimously.

The documents tabled and produced at that General Assembly are available at:

http://www.ada-france.org/article108.html

As of May 27, 2004, the Board of the Ada-France Association is thus comprised of the following persons:

Agusti Canals

Éric Forterre

Fabrice Kordon

Laurent Pautet

Thomas Quinot

Samuel Tardieu

As of May 28, 2004, the President Office of the Association is comprised of the following persons:

Laurent Pautet: secretary

Thomas Quinot: president

Samuel Tardieu: treasurer.

Take a look at http://www.ada-france.org/article2.html for further information.

Jun 10-17 - Open Source GIS Conference

From: James E. Hopper <hopperj@macconnect.com>
Date: Sun, 11 Jul 2004 05:07:20 GMT
Subject: Open Source GIS
Newsgroups: comp.lang.ada

Slashdot today has some pointers to the results of this years Open Source [Geographic Information System] conference

http://www.omsug.ca/osgis2004/proceedings.html

and

http://www.maptools.org/dl_scripts/redirector.php?path=omsug/osgis2004/2004-05-OSS-Briefing.doc).

One of the products they talk about is Ossim (http://ossim.org) which is a very nice open source package for supporting GIS efforts. While there is a QT GUI that works well on Mac/Linux/Windows (and now has very nice Mac and Windows installers), the real power is in the library that lets you build really powerful apps with very little code. For instance I wrote a nice drag and drop app in carbon that lets you convert a number of formats for things like digital terrain elevation data (DTED, DEM), National Imagery Transmission Format (NITF), etc into other formats like jpeg, etc in a couple hundred lines of code most of which is the GUI code.

I have done an Ada binding to the main parts of the library and while I haven't put it out yet on my website, I would be willing to share it with anyone who is interested in this type of thing. I encapsulated the Ossim C++ classes inside Ada tagged types and I would like some feedback on if this is the right way to do bindings to C++ classes. I have also converted a number of the Ossim Tutorial apps to Ada as examples of using the class. I have used it in a number of apps for work as well, but I cant really share those.

From: James E. Hopper <hopperj@macconnect.com>
Date: Sun, 11 Jul 2004 19:02:59 GMT
Subject: Re: Open Source GIS
Newsgroups: comp.lang.ada

Jeff C wrote:

> I am sort of interested. No specific applications but it would be interested to look at. Perhaps you can get a page at www.adaworld.com for it (or setup a sourceforge project)?

I am working with the ossim folks to add it to the baseline ossim stuff as soon as they look at the C++ code I added to ossim to support it. I was just looking for someone who is familiar with GIS or Ada tagged types to give me some feedback on the bindings before I put them out for the entire world. If this doesn't happen the bindings will get posted on the ossim site (ossim.org) in a couple of weeks.

Jun 14-18 - Ada-Europe 2004 Conference

From: Dirk Craeynest <dirk@heli.cs.kuleuven.ac.be>
Date: 3 Jun 2004 22:43:59 +0200
Organization: Ada-Europe, c/o Dept. of Computer Science, K.U.Leuven
Subject: Press Release - Reliable Software Technologies, Ada-Europe 2004
Newsgroups: comp.lang.ada, fr.comp.lang.ada

Final Call for Participation

9th International Conference on Reliable Software Technologies - Ada-Europe 2004

14 - 18 June 2004, Palma de Mallorca, Spain <http://www.ada-europe.org/conference2004.html>

*** Final Program available on conference web site ***

*** Printed proceedings available ***

*** Check out the tutorial program! ***
*** Register now! ***

Press release:

Conference on Reliable Software Technologies

International experts meet in Palma de Mallorca

Palma de Mallorca (3 June 2004 21:00) - The University of the Balearic Islands, sponsored by Ada-Europe and in cooperation with ACM's Special Interest Group in Ada and with Ada-Spain, organizes this year the "9th International Conference on Reliable Software Technologies - Ada-Europe 2004" from 14 to 18 June in Palma de Mallorca, Spain.

The conference offers a technical program, an exhibition, and several tutorials. There's a special session on "Ada 2005", further main topics are "Critical Systems Modeling", "Distributed Systems", "Real-Time Systems", "Scheduling", "Static Analysis", "Testing", etc.

Invited lectures by internationally renowned experts on the topics "Fixing Software Before It Breaks: Using Static Analysis to Help Solve the Software Quality Quagmire", "Benefits and Problems of Formal Methods", "On the Role of Conceptual Schemas in Information System Development", and "Can Middleware Be Reliable?" complete the program.

The conference takes place at the Palas Atenea hotel close to the historical city center of Palma. Registration is still open. The full "Advance Program" is available on the conference web site <http://www.ada-europe.org/conference2004.html>.

Contact: llamosi@uib.es (Albert Llamosí, conference chair)

Latest updates:

- The "Final Program" is available on the conference web site <http://www.ada-europe.org/conference2004.html>.

- The proceedings published by Springer are ready and will be distributed at the conference. More info is available at <http://www.springeronline.com/3-540-22011-9>. Abstracts can be checked out via the conference web site.

- Registration fees are very reasonable and the registration process is easy: fill out the 1-page form and fax it to the conference secretariat. Don't delay! <http://dmi.uib.es/~AE2004/documents/g_registration.pdf>

- Check out the 8 tutorials in the advance program and at <http://dmi.uib.es/~AE2004/AE2004tutorials.html>.

- For the latest information consult the conference web site.

July 6-10 - Libre Software Meeting

From: Lionel Draghi <Lionel.Draghi@Ada-France.org>
Date: Thu, 01 Jul 2004 23:53:41 +0200
Subject: [Ann] Ada on the 2004 Libre Software Meeting
Newsgroups: comp.lang.ada

The 5th edition of the Libre Software Meeting (http://lsm2004.abul.org/rubrique2.html) will take place from the 6th to the 10th of July 2004 in Bordeaux (France).

Within the technical topic "Very-high level programming languages for writing applications" (http://lsm2004.abul.org/article17.html), Ada-France organize the Ada presentation the 8th of July.

Those presentations will demonstrate not only how well suited is the language for free software development, but also how easy it is to install and learn.

* 13:00, /Lionel Draghi/ : opening

* 13:10, /Ludovic Brenta/ : language overview

* 14:05, /Jean-Pierre Rosen/ (Adalog <http://www.adalog.fr/>) : Ada, a language of choice for free software

* 14:45, Thomas Quinot (ACT-Europe <http://act-europe.fr/>) : AdaCore role in GNAT use for free software

* 15:20, /Samuel Tardieu/ : Ada open language : using Ada with other languages such as Forth, Erlang, etc.

* 16:15, /Thomas Quinot/ (ACT-Europe <http://act-europe.fr/>) : PolyOrb <http://libre.act-europe.fr/polyorb/>, the schizophrenic middleware, an innovative research project developed in Ada 95, now industrialized, free software from the earliest stages of its development.

* 16:50, /Stéphane Rivière/ :

  o Introducing AIDE (Ada Instant Development Environment), all Ada for Windows ready to run on a CD

  o Martin and Xavier (13 years both) will explain us how they learn programming in Ada with AIDE

* 17:30, /Ludovic Brenta/ : Ada within Debian

More details (in French) on the Ada-France web site http://www.ada-france.org/article111.html

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Subject: Re: [Ann] Ada on the 2004 Libre Software Meeting
Date: Fri, 09 Jul 2004 19:23:11 +0200
Newsgroups: comp.lang.ada

Here is my report from the event; I arrived back in Brussels yesterday at midnight.

The Libre software meeting went as planned. There were maybe 1000 persons overall. There were a dozen booths from several associations and free software projects. On Thursday afternoon, attendants could choose between 11 conferences on various themes, or joining any one of 14 small rooms where informal discussions were being held, or software being developed. Richard M. Stallman was present and, as usual, delivered his speech about how bad software patents are, and he was of course surrounded by a swarm of admirers.

Anyway, at 13:00 there were about 30 people in the auditorium, not counting the speakers. The presentations turned out to be a 6-hour marathon, packed with really interesting stuff. At the end there were still 2 people in the audience, who had stayed the whole time and looked very serious about learning Ada.

As an introduction to my speech about Ada, I showed off a picture of the C-130J cockpit filled with state-of-the art avionics, and this immediately captivated the audience. After my speech, I passed around the dozen or so brochures I'd brought from Barco Avionics. They also made quite a strong impression on the audience, and none remained in the room at the end of the day. I should have brought twice as many brochures.

I was most impressed by two 13-year-old youths who started learning programming in February this year, and are already Ada die-hards after playing with Python for a while, and also looking at Lisp, C and Java. They understand that Ada is not a fashionable language but still prefer using a good language than a fashionable one. Even more stunning, they even prefer using Emacs instead of more graphical IDEs such as GPS! They've written a 2000-line text-mode application in Ada that allows them to draw pictures using ASCII block characters, save them into text files, read back and display them. They designed the file format themselves, and it turns out it is quite similar to XPM.

They have a second application that uses these files to display a "Start" menu with a number of applets, one of which is a fully working calculator. The father of one of these youths, Stéphane Rivière of AIDE fame, taught them the basics of Ada during 45-minute courses on Sundays, and they did all the rest by themselves with very little supervision. After only 4 months since their first exposure to programming, they understand and routinely use separate compilation and encapsulation, and were asking me questions about multitasking and game programming in Ada!

My conclusions about the event:

- I was thrilled to meet some prominent Ada people whom I knew only by email. Lionel Draghi, Samuel Tardieu, Jean-Pierre Rosen, Thomas Quinot and Stéphane Rivière and his "MX" team: thank you!

- I was also thrilled to meet people I'd never heard of before, but who were appreciative and said so.

- The audience was small but we definitely carried our point across ("use Ada"). I'm pretty sure we made at least two converts, and probably more who remained silent or left before the end of the 6-hour "marathon".

- Areas for improvement include better posters and signs directing people to the auditorium; better speaker discipline (some speeches went overtime); and more time dedicated to questions and answers.

- There is so much we can say about Ada and free Ada software that we really have to make difficult choices in the topics we discuss, or else increase the length of time allocated to speeches.

This weekend, I will convert my slides to PDF and post them on the Libre Software Meeting web site[1]. They will also be mirrored on the Ada-France[2] web site, and probably on other sites as well. Good news is I now have them in both French and English versions; they may and will of course be reused in future events. To encourage this, I will license them under the GPL.

The next big event will be the FOSDEM[3], held in Brussels every year in February. The dates for 2005 have not been published yet, but I expect that it will take place during the weekend of 25-26 February 2005, take or leave a week. I plan to organise a similar session about Ada as part of this event. In the past two years when I attended this event, the audience was in the range of 3000 to 5000 developers, geeks, enthusiasts, and members of academia.

If anyone is willing to give a speech at the FOSDEM, please contact me.

[1] http://libresoftwaremeeting.org

[2] http://www.ada-france.org

[3] FOSDEM, Free and Open-Source Developers' Meeting, http://www.fosdem.org

Ada and Education

SPARK in universities

From: Rod Chapman <rod.chapman@praxis-cs.co.uk>
Subject: Improving Ada's image - Was: 7E7 Flight Controls Electronics
Date: 31 May 2004 02:22:22 -0700
Newsgroups: comp.lang.ada

Richard Riehle wrote:

> As to other universities, there is currently no incentive for professors to carry out any kind of research that might favor Ada. The scholarly journals prefer contributions that avoid Ada, the textbook publishers edit out excessive references to Ada, and the DoD is not funding any Ada research projects anymore.

We have recently been rather successful (by our own modest standards) at getting Ada into University programmes … How? The answer is simple: SPARK!

SPARK offers faculty some really interesting stuff to play with: design-by-contract, program proof, static analysis, hard real-time, embedded etc. etc. Secondly, the professional toolset is free (as in no dollars) to university faculty.

Off the top of my head, the following US institutions have taught SPARK on advanced courses in the last couple of years: NYU, Northern Iowa, Virginia, JMU, Oakland, USAFA, and NPGS. Not a bad start. Several others use SPARK in research: MIT springs to mind, plus York, Birmingham, Heriot-Watt in the UK.

There's also a pretty darn good textbook about SPARK, which helps. :-)

At SIGCSE this year, the SIGAda and ARA people were cleaned out of SPARK book samplers and CDROMs - we sent over 50 sets and could have probably shifted double that. We have subsequently been contacted by other authors who want to include SPARK in next editions of their programming languages textbook. Not a bad sign.

As for publications, we have some success - IEEE Transactions on SE, IEEE Software, CrossTalk (a DoD sponsored journal!) hardly seem like small-fry. Heriot-Watt's current work on automated proof is appearing in significant conferences and (soon) journals.

Finally, the _really_ great thing about SPARK is that you can tell your conference program committee, journal referees, and board of studies all about it, WITHOUT MENTIONING THAT IT'S Ada! (at least not until the light-bulb has gone on and people have realised how great it is...) This really does work - honest.

So here's a challenge for the Ada community: get SPARK into your local university CS program, without telling 'em it's Ada. Go ahead - you know you want to! When someone asks what the language is like, tell 'em it's hard-real-time-Eiffel-on-steroids. When someone asks which compiler you use, tell 'em "GCC of course"...go on give it a go - Praxis will support your efforts...

[For more information about SPARK Academic Support Programme see http://www.praxis-cs.co.uk/sparkada/universities.asp --su]

Ada in colleges and universities

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Mon, 07 Jun 2004 19:27:00 GMT
Subject: Re: Ada in colleges and universities.
Newsgroups: comp.lang.ada

Marius Amado Alves wrote:

> But there's no Ada being taught in Europe either. [...]

Glasgow University uses Ada 95 as the foundation language for CS1 and CS2, and has done since 1996.

FP using Haskell is added in CS2; C and Java are added in CS3 (it's a 4-year program.)

When certain senior professors tried to have Ada replaced with Java, the class head of CS1 threatened to resign. Ada 95 was retained. In 2001 there was a curriculum review, at which Java was again touted. Thanks to the staff teaching Ada, there was again no change of policy.

From: Jano <402450@cepsz.unizar.es>
Date: 8 Jun 2004 02:53:27 -0700
Subject: Re: Ada in colleges and universities.
Newsgroups: comp.lang.ada

Another counterexample: University of Zaragoza, Spain. Language of choice for several courses (at least four that I can remember, and tangentially used in several more).

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: Tue, 08 Jun 2004 00:03:26 +0200
Subject: Re: Ada in colleges and universities.
Newsgroups: comp.lang.ada

Peter C. Chapin wrote:

> You are probably talking about a computer science program, true? VTC offers computer engineering technology. It's more like computer engineering which, in turn, is more like electrical engineering than it is like computer science. Our students get a lot of hardware courses, including basic (and not so basic) electronics. The programming we are preparing them for is low level stuff... device drivers, embedded systems, specialized executives, etc. We tend to be relatively light on abstract theory and heavy on the "practical" construction of systems. If I suggest that we reintroduce Ada in the curriculum I know the question will be: "How much Ada is being used, relative to C, in the application domains we focus on?" How should I best answer a question like that?

At Barco Avionics, it is roughly 85% Ada, 15% C. The C is used for some microcontrollers not supported by the current Ada compiler, but I suspect there must be a way to use Ada there too.

From: Bill Findlay <findlaybill@blueyonder.co.uk>
Date: Mon, 07 Jun 2004 22:17:35 GMT
Subject: Re: Ada in colleges and universities.
Newsgroups: comp.lang.ada

Björn Persson wrote:

> As a side note, one of the books we had in the first course was Programming Language Concepts and Paradigms by David A. Watt. The examples of exception handling in Ada in that book made me decide that I had to have a closer look at Ada, and here I am now.

Then you might like to have a look at his new opus, "Programming Language Design Concepts", which, despite the publisher's blurb downplaying Ada, makes even better use of it for significant examples.

(I wrote the concurrency chapters in the new book as well. 8-)

BAe trains non-ITers as coders

URL: http://www.computerweekly.com/Article106744.htm
Date: Sat, 15 May 2004 17:47:10 +0200
Subject: BAe trains non-ITers as coders

IT Management: HR & Skills

by Bill Goodwin

Thursday 11 October 2001

BAe trains non-ITers as coders

British Aerospace is overcoming a shortage of skilled software engineers by hiring staff with little or no IT experience and training them in-house.

BAe's Combat and Radar Systems Division, which is based at four sites in the South of England, is taking on 50 staff a year with backgrounds in everything from physics to medieval history. Richard Knowles, engineering trainer at BAe, said, "We certainly could go out there and try to recruit software engineers but it is very difficult. Our approach is to bring

very bright people in and give them grounding in software engineering."

BAe uses a rigorous aptitude test to select candidates with the right thinking patterns to make good software engineers. "Our experience is that some people who don't realise they are good programmers do very well on the test," said Knowles. The company said it is able to train raw recruits in the basics of software engineering in just four weeks. The course, run by training company Global Knowledge, alternates programming theory with practical exercises so that candidates can immediately apply the principles they have learned. The students then undertake a one-week practical project.

Although the candidates learn the Ada programming language, which is widely used in the defense industry, the aim is to teach software engineering principles rather than Ada programming. Recruits are able to start work on real projects as soon as their four weeks' training is complete, said Knowles

Ada-related Tools

Ada0Y packages for Ada95

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Mon, 17 May 2004 22:58:07
Subject: First release of new Ada.Calendar. packages (AI-351)*
Newsgroups: comp.lang.ada

Developed on WinNT/WinXP, should be portable to any Ada95 compiler but was developed using GNAT 3.15p and ObjectAda v7.2.2 (U13).

See http://www.martin.dowie.btinternet.co.uk/

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Mon, 26 Jul 2004 21:44:05
Subject: [Ann] More Ada0Y packages for Ada95!
Newsgroups: comp.lang.ada

I've updated my web page to include the following implementations:

1) AI-248 - Ada.Directories.*

2) AI-351 - Ada.Calendar.*

3) AI-301 – Ada. (Wide_)Text_IO. (Wide_)Unbounded_IO

4) AI-302 - Ada.Containers.*

5) AI-296 - Ada.Numerics.Generic_ [Complex|Real]_Arrays

6) AI-346 - Ada.Numerics.Generic_ [Complex|Real]_Arrays.Generic_Roots

The implementation of Ada.Containers.* is a 'back-port' to Ada95 of Matthew Heaney's reference implementation available from: http://charles.tigris.org/. This version has a couple of extra explicit access-to-subprogram types that will be replaced with something better for Ada0Y but are otherwise identical.

Item 6 above is really only 'adabrowse'-able just now but if I get time I'll actually implemented it! :-)

[See also "Ada0Y.Directories - AI-248 Implementation" in AUJ 24-3 (Sep 2003), p.138. --su]

Command Line Argument Packages

From: Jeff C <jcreem@yahoo.com>
Date: Tue, 27 Jul 2004 03:44:09 GMT
Subject: Command Line Argument Package
Newsgroups: comp.lang.ada

I've posted a command line argument helper/utility package that I've had around for a while. I've used it on a few small things and had intended to clean it up some more and post it someday but it is already pretty useful.

http://newserver.thecreems.com/article.php?story=20040726220542693

The purpose of this package is to provide a cleaner way (or at least a more fun way) to access command line arguments and to help automate the generation of help page for the program.

There is some documentation in the header and a simple test program that demonstrates some of the concepts of the package. If anyone finds it useful I'll write up some more complete docs.

It is a different approach than the GNU Getopts library (and the Similar Ada version at http://www.adapower.com/reuse/). If you are looking for a more traditional getopt approach, look there.

From: Jeff C <jcreem@yahoo.com>
Date: Tue, 27 Jul 2004 23:50:57 GMT
Subject: Re: Command Line Argument Package
Newsgroups: comp.lang.ada

Larry Kilgallen wrote:

> Is this restricted to the "-this -THAT" UNIX-style command lines?

It is "designed" around the UNIX/GNU approach (-f hello.txt or --file hello.txt) but it really does not make the -this assumption. Will work just fine with /X /A /C style "Win32" arguments but it will then not accept win32 style subarguments (since they are often concatentated onto the argument in question (e.g. format /T:80)

Nothing stops you from using UNIX like options on win32 platforms. Plenty of programs do.

Granted on vxWorks it probably won't work for calling arbitrary routines from the target shell.

On an osless non-command-line OS it might not work at all.

From: Björn Persson <rombo.bjorn.persson@sverige.nu>
Date: Tue, 27 Jul 2004 14:27:57 GMT
Subject: Re: Command Line Argument Package
Newsgroups: comp.lang.ada

Nice package. I see you too have some of the ideas I have that have made me write my parameter handler Orto, which will be a part of AdaCL. Like your package, Orto looks up parameters by name rather than position, and prints help texts. It also interprets the parameters as numbers, Booleans and so on. Its command line syntax is somewhat different from yours.

Orto isn't quite finished yet and the documentation is mostly missing, but it's functional on Gnu-based systems. The code can be viewed at http://cvs.sourceforge.net/viewcvs.py/adacl/adacl/Include/
(the adacl-command_line-orto* files).

Auto_Text_IO & SAL

From: Stephen Leake <stephen_leake@acm.org>
Date: 07 Aug 2004 12:22:56 -0400
Subject: new SAL, Auto_Text_IO releases
Newsgroups: comp.lang.ada

SAL version 1.70 is released.

Auto_Text_IO 3.03 is released.

SAL is my Ada library; it has containers, spacecraft math, config files, other misc stuff. New in this release is a small collection of GtkAda widgets, some of which work :). More importantly, there is a test harness for automating GUI tests. Also misc fixes in other SAL packages.

Auto_Text_IO is an ASIS-based tool that generates Text_IO packages for types in Ada packages; very useful for persistent storage. New in this release: support for Standard.Duration, Standard.Character, and miscellaneous other fixes.

For full information, and to download the latest versions, see my web page http://www.toadmail.com/~ada_wizard/

[See also same topic in AUJ 24-4 (Dec 2003), p.210 and also "SAL 1.60" in AUJ 25-2 (June 2004), p.48. --su]

A# and Mono

From: Jeff <jeff.huter@bigfoot.com>
Date: 1 Jul 2004 18:29:13 -0700
Subject: A# and Mono
Newsgroups: comp.lang.ada

[For more information about A# see "A# - Port of GNAT to .NET" in AUJ 24.1 (March 2003) 24-25. --su]

Has anyone tried A# with Mono? From the information listed on A# site http://www.usafa.af.mil/dfcs/bios/mcc_html/a_sharp.html, it would seem that A# only worked with .Net on Windows. But it may mean, the author has only tested on Windows.

If A# worked on Mono without trouble, it should provide a nice cross platform de-

velopment environment for Ada. This also should address one of the biggest weaknesses with Ada development -- up to date development frameworks.

From: Brice Carpentier <brice@daknet.org>
Date: Fri, 02 Jul 2004 12:57:27 +0200
Subject: Re: A# and Mono
Newsgroups: comp.lang.ada

I contacted the developer, and it seems that this should be possible since this is mostly written in C#, but he wasn't sure.

I must add I also think having this project work on Mono could be determinant in democratizing Ada.

ECLAT - Open Source Ada 2005 Compiler

From: Nick Roberts <nick.roberts@acm.org>
Date: Wed, 14 Jul 2004 01:56:18 +0100
Subject: Re: Ada 2005 Grammar
Newsgroups: comp.lang.ada

[...] I'm happy to say that I'm making pleasingly good progress on the parser for ECLAT [Experimental Compiler Library And Toolset]. I've only just started a project for it on SourceForge. If you go to

http://sourceforge.net/docman/?group_id=113935

There is a modest amount of introductory material to read.

How to compile GCC 3.4

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Thu, 01 Jul 2004 10:29:50 +0200
Subject: Compiler gcc 3.4.1 How To
Newsgroups: comp.lang.ada

I have written a little How To on compiling the gcc 3.4.1:

http:// ada.krischik.com/gnat-3_4_1.html

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Tue, 27 Jul 2004 12:00:25 +0200
Subject: compile gcc 3.4.2 How To
Newsgroups: comp.lang.ada

I have updated the How To on compiling the gcc 3.4.1 - and its now gcc 3.4.2:

http://ada.krischik.com/gnat-3_4_2.html

And also now with ASIS and XML/Ada. What's a compiler worth without libraries.

Ada and the new GCC

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: 17 May 2004 07:39:44 GMT
Subject: Re: Ada and GCC 3.5 tree-ssa
Newsgroups: comp.lang.ada

Ian S. Nelson wrote:

> I know this has probably been answered but what's the plan with GCC 3.5 regarding Ada?
> Is someone working on the tree-ssa modifications to the Ada front-end or is that still out standing?

My understanding is that Ada Core Technologies does plan to update the Ada front-end for tree-ssa. However, as for any other company, their priorities are set by their customers, so it is not known just when they will find the time to do the update. This is a large change to the Ada front-end. As a prerequisite, it must support function-at-a-time compiling.

The tree-ssa branch has been merged into the main line, and the announcement [1] says that this broke the Ada and Fortran 77 front-ends. This was foreseen, as evidenced by the various emails that I reproduce in the Ada policy for Debian [2]. (BTW, I need to update that document).

[1] http://gcc.gnu.org/ml/gcc/2004-05/msg00679.html

[2] http://users.skynet.be/ludovic.brenta/debian-ada-policy.html

Symbolic Multinomial Algebra

From: Tom Moran <tmoran@acm.org>
Date: Mon, 31 May 2004 22:56:58 GMT
Subject: Ann: symbolic multinomial algebra
Newsgroups: comp.lang.ada

I couldn't find a simple, easy to integrate, symbolic algebra package, so I've posted one at [http://home.comcast.net/~twmoran/multinom.zip --su]

The sample program includes the Knuth 2.2.4 example:

```
type Vars is (x,y);
function Image(Var : Vars)
  return String is ...
package Polys is new
  Multinom (Vars, Image);
  use Polys;
  A : Polys.Multinomials :=
   X**4 + 2.0*x**3*y**1 +
    3.0*x**2*y**2 +
    4.0*x**1*y**3 + 5.0*y**4;
  B : Polys.Multinomials :=
   x**2 - 2.0*x**1*y**1 +
    y**2;
  ...
  Put_Line("Product is " &
   Image(A*B));
```

From: Craig Carey <research@ijs.co.nz>
Date: Thu, 03 Jun 2004 10:53:18 +1200
Subject: Re: Ann: symbolic multinomial algebra
Newsgroups: comp.lang.ada

I have an Ada 95 symbolic algebra program ('Tope') targetted to solve linear quantifier elimination problems such as this:

```
Inp: (Exists
   x)(a<x)(b<=x)(x<=c)(x<=d)
Out: (a<c)(a<d)(b<=c)(b<=d)
```

Online at: [http://www.ijs.co.nz/code/software.htm --su]

The Ada Yacc parser with UMASS extensions is used.

The existing REDLOG led into a breakdown in failing to simplify the symbolic output "Exists", and better could by using an operations research algorithm to compute dual polytopes thrice.

The dual polytope can be computed with the Chernikova algorithm of Mr LeVerge of 'irisa.fr', which is a small hard-to-port C program online here (at the Parma Polyhedra website): http://www.cs.unipr.it/ppl/Documentation/chernikova.c

(Mentioned on http://www.cs.unipr.it/ppl/Credits/)

It is curious how such a tiny 'satisfactory' C program can be at the centre of a lack of good progress of C programmers and also really quite hard to port to good Ada code.

The Italian research project is testing out the use of a C++ style for the syntax interface.

At Irisa (France), the Polka project uses real Ocaml for the front end which actually wouldn't compile for me: http://www.irisa.fr/prive/bjeannet/newpolka.html

The symbols are Real-valued. [...]

From: Craig Carey <research@ijs.co.nz>
Date: Thu, 03 Jun 2004 23:33:02 +1200
Subject: Re: Ann: symbolic multinomial algebra
Newsgroups: comp.lang.ada

[...] Whoops: that's disputable and vague over what 'progress' for dual polytope projects including: Cdd, Polka, Parma, http://www.cs.umd.edu/projects/omega/ ? [integers QE], Polylib, etc.

Cdd duals: http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html

My 'hard to port' seems a bit incorrect and it was about cutting 2 C arrays to leave 4 Ada matrices.

AWS 2.0p - Ada Web Server

From: Pascal Obry <pascal@obry.org>
Date: 09 Jun 2004 10:38:31 +0200
Subject: AWS 2.0p released
Newsgroups: comp.lang.ada,fr.comp.lang.ada

AWS 2.0p is out. Be sure to have a look at the new ada2wsdl tool (based on ASIS) that makes quite easier to build Web Services using AWS. Note also that AWS CVS repository has been opened for read-access.

Here is a copy of the readme.txt.

AWS - Ada Web Server

2.0 release / SOAP 1.2

We are happy to announce the availability of the AWS 2.0 release. The API could change slightly at this stage but should be fairly stable now.

AWS stand for Ada Web Server. It is a small yet powerful HTTP component to embed in any applications. It means that you can communicate with your application using a standard Web browser and this without the need for a Web Server. AWS is fully developed in Ada with GNAT.

AWS support SOAP/WSDL, Server Push, HTTPS/SSL, client HTTP, hotplug modules... We have worked very hard to make this release as stable as possible. Note that Hotplug modules are very nice but have a potentially security hole as it is implemented today. A new secure implementation will be proposed in a future version.

AWS comes with SOAP/WSDL support, two tools are proposed:

ada2wsdl which generates a WSDL document from an Ada spec

wsdl2aws which generates stubs/skeletons AWS code from a WSDL document

Both tools have mapping for standard Ada types but also supports Ada enumerations, character, records and arrays.

The SOAP implementation has been validated on http://validator.soapware.org/.

Here are the main changes since AWS 1.4:

Notes:

You can have a look at docs/TODO file to see what are the topics that we will probably implement in future releases.

NOTE: Since we have switched to the .PNG file format we have found that Netscape Navigator is not able to display the PNG transparent layer properly!

The OpenSSL libraries (optional) distributed are for Windows only. On UNIX you'll have to build the libraries from sources, it is quite easy to do so. This has been tested on GNU/Linux without trouble.

The LDAP binding will use the LDAP dynamic library on Windows. On UNIX you need to build and install OpenLDAP.

See documentation for build information.

Validation:

AWS 2.0 has been compiled and has passed all tests on:

Windows XP, GNAT 3.15a1, 3.16a1, 5.01a and 5.02a

GNU/Linux x66, GNAT 3.16a1 and 5.02a

SPARC Solaris 8, GNAT 5.01a

HP-UX 11, GNAT 5.03w

Others platforms / compiler version combinations have not been tested, it does not mean that it's not working.

Previous versions of AWS have been built on FreeBSD 4.1 and MacOS X.

Pointers:

AWS User's Mailing List:

http://lists.act-europe.fr/mailman/listinfo/aws

AWS Home Page (sources and printable documentations in Postscript and PDF):

http://libre.act-europe.fr/aws

Templates_Parser sources:

Templates_Parser module (sources and documentation) is provided with AWS distribution. Version 7.5 is distributed with AWS 2.0.

Latest version of this module and the documentation can be found at:
http://perso.wanadoo.fr/pascal.obry/contrib.html
http://perso.wanadoo.fr/pascal.obry/templates_parser.html

Templates_Parser is a very useful add-on for AWS. You should have a look at it if you plan to develop a Web application. Templates_Parser permits to completely separate the HTML design from the Ada code.

Some other Templates engines are WebMacro, FreeMarker, PHP, ASP, JSP and Velocity. All of them are based on explicit iterators (#foreach with a variable) where Templates_Parser is based on implicit ones (you use a more intuitive table iterator). Be sure to check the documentation. Only the Velocity project has the goal to support complete separation of HTML design and code.

GNU/Ada - GNAT

You need at least version 3.15 to build and use AWS 2.0.

http://libre.act-europe.fr/GNAT/

Socket binding (Optional):

Since AWS 1.2 you need at least version 1.0 of the Socket binding. Note that by default AWS uses GNAT.Sockets.

for Win32:
http://perso.wanadoo.fr/pascal.obry/contrib.html
http://vagul.tripod.com/adasockets.tgz

for UNIX:
http://www.rfc1149.net/devel/adasockets

XMLada (optional):

You need this library only if you want to use AWS SOAP feature. You need at least XMLada 1.0.

http://libre.act-europe.fr/

POSIX Binding (optional) :

for Win32:
http://perso.wanadoo.fr/pascal.obry/contrib.html

for UNIX:
http://www.cs.fsu.edu/~baker/florist.html

OpenSSL library (optional):

Sources for UNIX or Win32:
http://www.openssl.org

binaries for Win32:

Included with the main AWS distribution (win32 directory).

Note that we have used and we distribute (for Win32 platform) OpenSSL version 0.9.7c with this AWS release. OpenSSL has been built with GNAT 5.01a C subsystem (based on GCC 3.2.3) with -O3 optimization level.

See OpenSSL license (docs/openssl.license).

OpenLDAP library (optional):

Sources for UNIX or Win32:
http://www.openldap.org/

binaries for Win32:

Included with the main AWS distribution (win32 directory). The import library will bind to the Microsoft LDAP dynamic library.

Windows Services API (optional):

To build the runme demo as a Windows NT/2000 services you must download the services API made by Ted Dennison for his SETI@Home project.
http://www.telepath.com/~dennison/Ted/SETI/SETI_Service.html

License:

AWS is distributed under the GMGPL (GNAT Modified GPL) license. This license ensures that commercial applications can be built using AWS. Note that AWS comes with a set of components. Those components are using a license compatible with the AWS's one. For information about component's individual licenses see include/readme.txt.

[See also "AWS 1.4 - Ada Web Server" in AUJ 25-1 (March 2004), pp.8-9. --su]

From: Pascal Obry <pascal@obry.org>
Date: 15 Jun 2004 23:10:08 +0200
Subject: Re: AWS 2.0p released
Newsgroups: comp.lang.ada, fr.comp.lang.ada

Volkert wrote:

> Now I do. We have interest to use AWS for Web Services in our system. That is why I asked ... ACT will give support for AWS I think ;-)

That's right, and this is a very important point for some projects.

PGAda - Ada PostgreSQL binding

From: Lomere <vaval_1@hotmail.com>
Date: 12 May 2004 05:42:39 -0700
Subject: Ada and Oracle
Newsgroups: fr.comp.lang.ada

[Translated from French.] I would like to write an Ada program that connects to an Oracle database and which issues SQL queries. What utilities do I need (libraries, documents …)? Thanks in advance for any advice.

From: Lionel Draghi <Lionel.Draghi@fr.thalesgroup.com>
Date: Wed, 12 May 2004 14:53:51 +0200
Subject: Re: Ada et Oracle
Newsgroups: fr.comp.lang.ada

Excerpts from the FAQ of the forum hosted at http://www.rfc1149.net/fcla/

3.1.4 How to access a database from Ada?

The GNADE project, freely available at: http://www.sourceforge.net/projects/gnade aims to provide access from Ada to all sorts of databases.

Furthermore, a library specifically tailored for PostgreSQL is freely available at: http://www.rfc1149.net/devel/pgada.

Mneson - Database System

From: Marius Amado Alves <amado.alves@netcabo.pt>
Date: Tue, 1 Jun 2004 16:56:16 +0100
Subject: Mneson announcement and help request
Newsgroups: comp.lang.ada

Mneson is a developing 100% Ada database system. Latest version 20040601. http://www.liacc.up.pt/~maa/mneson

The core seems well, but there's a problem in an auxiliary module that I'm having difficulty analysing due to an apparent gprof bug. So I appreciate the help of anyone interested in making Ada the next generation database technology :-)

An excerpt from "revision_history.txt" follows. Please contact me on any issue. Thanks a lot. [...]

[See also same topic in AUJ 25-2 (Jun 2004), pp.54. --su]

From: Marius Amado Alves <amado.alves@netcabo.pt>
Date: Wed, 02 Jun 2004 12:40:54 +0100
Subject: Re: Mneson announcement
Newsgroups: comp.lang.ada

Wojtek Narczynski wrote:

> I have two licensing questions:
> - Have you managed to make any money with this license? Do you know of any such case?

No, not yet. (I'm not particularly happy with the current form of the license. There are other forms of the same philosophy. Explore the SDC references. The philosophy is utterly simple: it's open source, and if the user ever makes money, then the author gets a cut. A number of people seem to find this the most sensible open source philosophy. Ask Bob Leif how it can be implemented... and stand back :-)

> - Isn't your code, by accident, GPL?

No.

> AI-302 reference implementation seems to be such license.

God's in the details. AI-302 is like GMGPL (GNAT-Modified GPL), which is equivalent to LGPL, not to GPL. You can instantiate the generic units like crazy.

Also note that:

- Ada.Containers will take the place of AI302 soon

- Mneson does not desperately depend on any of these

From: Marius Amado Alves <amado.alves@netcabo.pt>
Date: Sun, 06 Jun 2004 14:21:56 +0100
Subject: [Ann] Mneson Manual
Newsgroups: comp.lang.ada

Released Mneson Manual version 20040606, revised and consolidated.

http://www.liacc.up.pt/~maa/mneson

GNADE 1.5.3a - GNAT Ada 95 Database Development Environment

From: Michael Erdmann <Michael.Erdmann@snafu.de>
Date: Mon, 31 May 2004 19:48:21 +0200
Subject: Release of GNADE 1.5.3a
Newsgroups: comp.lang.ada

I like to announce the latest release (1.5.3a) of the GNU Ada Database Development Environment (GNADE).

The release features a common build procedure for Cygwin, *nix and Windows (DOS) and some corrections to the MySQL bindings.

For further details pls. refere to http://gnade.sourceforge.net/

[See also "GNADE 1.5.2 - GNAT Ada 95 Database Development Environment" in AUJ 25-2 (Jun 2004) p.54 --su]

Semtools - Semantic Tools for Ada

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Wed, 26 May 2004 10:56:40 +0200
Organization: Adalog
Subject: New version of Semtools/Adasubst 1.3
Newsgroups: comp.lang.ada

I've just released a new version of semtools (actually, only Adasubst has changed).

New feature:

You can give a substitution directly on the command line in place of the dictionary file. This makes it easier when you want to just change a single identifier in your program:

adasubst "pack.bad_identifier => Good_Identifier" my_main -r -o temp/

will change every occurrence of "bad_identifier" declared in package "pack" to "Good_Identifier" into unit My_Main (and recursively in all units My_Main depends on), putting changed units into directory temp/

New mode:

adasubst -R is a new mode of Adasubst that will remove all representation clauses from the specified units. Useful to check whether representation clauses have an impact on the efficiency of your program.

No bug fix - no bugs were reported on previous version.

As usual, you can download Semtools from Adalog's components page http://www.adalog.fr/compo2.htm. (sources and ready-to-use executables (gnat 3.15p) for Linux and Windows). Alternatively, the sources are part of the ASIS-for-GNAT project on Sourceforge.

[See also "Adasubst & Adadep" AUJ 25-2, "Adasubst & Adadep - Semantic Tools for Ada" in AUJ 24-1]

Emacs mode for GNAT project files

From: Rolf Ebert <rolf.ebert@gmx.net>
Date: 2 Jun 2004 23:44:19 -0700
Subject: [Ann] Emacs major mode for editing GNAT project files
Newsgroups: comp.lang.ada

Please see below an Emacs mode for editing GNAT project files ending in .gpr. It reuses almost all from the standard Ada-mode with the following differences:

- add font-locking for "project" and "external"

- correctly indent after a "project Foo is"

It would have been shorter (in number of bytes) to integrate gpr support into ada-mode.el as a variant, but it was easier to write a new file.

[Get the code from the original post in comp.lang.ada --su]

HTH

Rolf

[See also "GNU Ada-mode for Emacs 21" in AUJ 24-4 (Dec 2003) p.208. --su]

Package Debug V2.3

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 22 Jul 2004 11:21:37 +0200
Organization: Adalog
Subject: Package Debug V2.3
Newsgroups: comp.lang.ada

Adalog announces a new release of package Debug.

It adds new facility for tracing tags (useful when you get a Constraint_Error on a tagged type conversion, and you wonder what the bloody type of the thing may be...), and a trace with a counter that you can put in a loop to see how many times you go through the loop.

Still no bugs ever reported.

As usual, it is licensed under GMGPL, and is available from http://www.adalog.fr/compo2.htm

[See also "Package Debug V2.2" in AUJ 25-2 (Jun 2004) p.58. --su]

SC_Timer - Session Chair Timer

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Thu, 5 Aug 2004 14:21:46 +0200
Organization: Adalog
Subject: SC_Timer, the Session Chair Timer V1.0
Newsgroups: comp.lang.ada

SC_Timer is a kind of count-down timer which is especially convenient for session chairs in conferences who need to manage their speakers. It offers a big white count-down display that you can make as big as your screen, for the speaker to see. It will turn yellow -say- ten minutes before time is over, then turn red five minutes before the end. When the presentation time is over, it will display a big red flashing "Off". Alternatively, the timer can be set to count up (from 0 to programmed time) instead of down.

There is also a regular clock in the upper right corner that tells you the current time.

Of course, the program is really a general count-down timer, and you can use it for all your count-down needs!

SC_Timer is fully written in Ada. It can be used as a not-too-complicated example of using Ada, GtkAda (the Glade project file is included in the source distribution), tasking, and tasking with GtkAda!

SC_Timer is available from Adalog's free programs page at http://www.adalog.fr/progs2.htm. It is distributed as source or ready-to-go executable for Linux and Windows.

Cheaptickets

From: Tom Moran <tmoran@acm.org>
Date: Tue, 13 Jul 2004 04:25:51 GMT
Subject: Q&D tool for cheaptickets
Newsgroups: comp.lang.ada

Writing even a quick and dirty program takes some time. This is a package to save some of that time. It process the results of one or more flight requests to www.cheaptickets.com, merging the results of multiple searches (on different flight dates, for instance) and, letting you assign a cost to red-eyes, early morning, more distant airports, etc, it produces a list sorted from best to worst. A sample driver with some parameters for planning a trip from Silicon Valley (SJC, SFO, and OAK airports) to the 11/2004 Atlanta SIGAda is included.

[Look for cheaptickets.zip at http://home.comcast.net/~twmoran --su]

PolyORB 1.1r

From: Laurent Pautet <pautet@dorine.enst.fr>
Date: Tue, 08 Jun 2004 22:41:46 +0200
Subject: PolyORB 1.1r
Newsgroups: fr.comp.lang.ada,comp.lang.ada

The PolyORB team is proud to announce the release of PolyORB 1.1r

(http://libre.act-europe.fr/polyorb/).

This release contains a CORBA-compliant instantiation of the PolyORB generic middleware. This release is not supported by ACT. Release 1.1r is a stable snapshot of the PolyORB generic middleware implementing the new features described below.

In addition to PolyORB 1.0p (2003-06-16), it includes:

* a significant increase in performance: from 30% up to 40% depending on the configuration,

* fixes for several bugs and memory leaks,

* extended support for CORBA and GIOP specifications,

* the PolyORB User's Guide,

* the MIOP/UIPMC protocol stack, Unreliable Multicast Inter-ORB Protocol, following the OMG standard,

* the DIOP protocol stack, Datagram-based Inter-ORB Protocol, a specialization of GIOP for oneway requests,

Other instantiations of PolyORB are available in the public PolyORB CVS repository for testing purposes. Available instantiations include DSA (Distributed System Annex), MOMA (Message Oriented Middleware for Ada) and AWS (Ada Web Server).

PolyORB is primarily developed by Jérôme Hugues, Thomas Vergnaud, and Laurent Pautet (Télécom Paris), and Thomas Quinot (ACT Europe). Fabrice Kordon (LIP6) also participates in the project. Vadim Godunko regularly contributes by submitting patches.

[See also "ACT - PolyORB 1.0p" in AUJ 25-1 (March 2004) pp.10-11 --su]

From: Laurent Pautet <pautet@dorine.enst.fr>
Date: Wed, 09 Jun 2004 11:18:58 +0200
Organization: ENST, France
Subject: Re: PolyORB 1.1r
Newsgroups: comp.lang.ada

Martin Krischik wrote:

> I take this an an indication that Glade is indeed dead.

Let's summarize the story.

1995

GLADE first and unique ORB for DSA (Distributed Systems Annex) done by a research team (mostly L. Pautet and S. Tardieu) at Télécom Paris with the help of CSC and then supported by ACT (at this time).

1999

AdaBroker a free ORB for CORBA done by the same research team at Télécom Paris (basically GLADE team, T. Quinot and other contributors).

2002

PolyORB a generic ORB (which provides DSA, CORBA, MOM, ...) has been designed by a research team from Télécom Paris, LIP6 and ACTE (mostly L. Pautet, T. Quinot, J. Hugues and F. Kordon). This middleware is based on GLADE and AdaBroker know-how. PolyORB is supported by AdaCore. It is also the result of the current research work on middleware architecture carried out by Télécom Paris/LIP6.

Many papers in non-Ada conferences.

GLADE is definitively alive. There are customer releases for 3.16 and 5.02. Unfortunately, there is no public release from a long time. The GLADE CVS tree may soon become available. Note that when GLADE is integrated in PolyORB, GLADE will not be dead. It will be part of a larger middleware. GNAT will probably still be the only Ada environment providing Annex E.

PolyORB is definitively alive. It is a successful professional ORB and more important it is a successful Ada research project in the free software area and in the middleware community (many papers in non-Ada conferences).

I can tell you it is a hard job to be part of the middleware community when you are using Ada, when you have to deal with C++ ORBs (even in the RT community) like TAO (intensively supported through DARPA projects) and when you are not part of the Java reflective bla approach.

So yes, we don't have a lot of time to communicate about GLADE, AdaBroker and PolyORB on the web or through press releases. But this is not really our job :) But well, we do our best to use Ada to do our research work. We communicate on it in conferences (therefore we promote Ada mostly outside the Ada community).

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Wed, 09 Jun 2004 13:44:11 +0200
Subject: Re: PolyORB 1.1r
Newsgroups: comp.lang.ada

I just had the impression that PolyORB would fully replace GLADE. I never meant to criticise your excellent effort.

GLADE CVS tree

From: Laurent Pautet <pautet@dorine.enst.fr>
Date: Fri, 11 Jun 2004 14:31:09 +0200
Organization: ENST, France
Subject: GLADE
Newsgroups: comp.lang.ada

GLADE CVS tree is available at http://libre.act-europe.fr/glade/

[See also "GLADE For GNAT: New Project" in AUJ 25-2 (Jun 2004) p.57. --su]

From: "Dr. Adrian Wrigley" <amtw@linuxchip.demon.co.uk.uk.uk>
Date: Fri, 11 Jun 2004 16:49:07 +0100
Subject: Re: GLADE
Newsgroups: comp.lang.ada

This is good news! Thank you Laurent and the GLADE team. [...]

I see from the NEWS file that PolyORB has replaced GARLIC. Can we choose the PCS? or do we need to download PolyORB as well? Is GARLIC now obsolete?

Laurent wrote:

> Note that when GLADE is integrated in PolyORB, GLADE will not be dead.

Has this integration happened yet? Or is that still a future development?

From the answers to the PolyORB 1.1r thread, I get the impression that GLADE has become the DSA "personality" of PolyORB. I'll try building it when I can get a CVS login, and when I understand better what I need to do. Hopefully it will work with GNAT 3.15p.

Ada-related Products

Praxis Critical Systems - SPARK Book Upgrade Packages Available

From: Rod Chapman <rod.chapman@praxis-cs.co.uk>
Date: 4 Jun 2004 08:49:15 -0700
Subject: SPARK Book release 7.1 upgrades...
Newsgroups: comp.lang.ada

We're pleased to announce the availability of release 7.1 of the SPARK Toolset for purchasers of the "SPARK Book" by John Barnes.

These are available now for Windows and IA32/Linux from www.sparkada.com

This release includes the SPARK Examiner version 7.1 and Simplifier 2.15. The latter includes significant improvements to its theorem proving tactics for VCs involving quantifiers and updates to composite objects (e.g. arrays).

The language and Examiner 7.1 support the recent "RavenSPARK" tasking features of SPARK.

Full documentation is included in each of the packages.

[See also "Praxis Critical Systems - SPARK Book Upgrade Packages Available" in AUJ 24-4 (Dec 2003) p.213 and "New SPARK Book - Sample Chapters On-Line" in AUJ 24-2 (Jun 2003) p.89. --su]

ObjectAda Update

From: Aonix Ada Support <adasupport@aonix.com>
Date: Thu, 10 Jun 2004 14:58:19 -0700
Subject: Intel-OA: New ObjectAda update
To: intel-objectada@aonix.com

A new patch update to ObjectAda for Windows 7.2.2 (1102) is now available. The update download file and the Release Notes are available at http://www.aonix.com/ada_patches.html

Please see the Release Notes for more information.

Ada and CORBA

CORBA vs. DSA

From: I R T <rambam@bigpond.net.au>
Date: Tue, 08 Jun 2004 09:10:44 GMT
Subject: Distributed programming, a killer application for Ada?
Newsgroups: comp.lang.ada

Ada has had concurrency support from the beginning.

People are now starting to realise that support for concurrency and distributed programming is a Good Thing.

Recent languages like Oz/Mozart, Cw, Erlang have support for this.

Perhaps demonstrating how to write concurrent distributed programs with Ada might help to create interest.

From: Wes Groleau <groleau+news@freeshell.org>
Subject: Re: Distributed programming, a killer application for Ada?
Date: Wed, 09 Jun 2004 21:57:10 -0500
Newsgroups: comp.lang.ada

Wes Groleau wrote:

> Nobody but ACT supports the distributed annex. Apparently they think that CORBA is better.

Laurent Pautet wrote:

> I am a little bit surprized by such a statement.

Would have been clearer had I said, "Apparently the others think that their customers are more interested in CORBA."

However, I saw a post that hinted one other vendor supports the distributed annex.

In my previous job, even the most gung-ho Ada fan thought CORBA was a better choice than Annex E for distribution, just because (like it or not) we were being forced to interface with Java and C and even C++

From: Nick Roberts <nick.roberts@acm.org>
Date: Thu, 10 Jun 2004 03:36:10 +0100
Subject: Re: Distributed programming, a killer application for Ada?
Newsgroups: comp.lang.ada

Warren W. Gay wrote:

> What gives rise to "apparently they think."? It is best not to read too much into impressions. They may in fact have customer demand for CORBA, but that doesn't necessarily mean that the distributed annex has fallen out of favour.

I'm certain that specific customer demand for CORBA is one big reason.

However, be aware that the DSA (Annex E) and CORBA are not direct alternatives for one another. It is a crucial aspect of CORBA that it supports programs developed in isolation from one another except for the interfaces between them (defined, actually or in effect, by OMG IDL); in particular, multiple programming languages can be used. The DSA requires that all the partitions of a distributed program are compiled in the same environment at least to the extent that every partition has a means of identifying the others it depends on; CORBA has no equivalent requirement. The DSA requires (but does not specify) a mechanism by which all the partitions come into execution simultaneously (in effect). CORBA makes provision (specifying some, but not all of of) for much more flexible mechanisms, for example supporting the execution of server programs upon demand. There are many other significant and fundamental differences.

(Incidentally, I intend to support /both/ DSA and CORBA in AdaOS, eventually.)

From: Tom Moran <tmoran@acm.org>
Date: Fri, 18 Jun 2004 22:16:16 GMT
Subject: Re: Distributed programming, a killer application for Ada?
Newsgroups: comp.lang.ada

I wrote a simple stub maker tool and system.rpc for Janus distributed programming without much difficulty.

I've posted this to [http://home.comcast.net/~twmoran/part.zip]

Look near the bottom. As usual, it's source, free, and no warranty.

Ada and Linux

Debian Policy for Ada

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: Sun, 11 Jul 2004 19:41:28 +0200
Subject: Announce: Debian Policy for Ada, First Edition
Newsgroups: comp.lang.ada

After receiving comments from several people, I have updated the Debian Policy for Ada. I deem it to be worthy of an official First Edition. This does not however mean that I will not accept any more comments or contributions; these are always welcome. Compared to Draft 4, here are the changes:

Introduction:

* More contributors acknowledged.

Choosing the GNAT compiler:

* New section: "History of GNAT".

* GNAT Pro: described the test suite used at ACT.

* GNAT Public releases: referred to the test suite.

* Timeline of GNAT Pro releases: mention the "wavefront" releases.

* Timeline of FSF releases: GCC 3.4.1 released.

* GLADE CVS repository at ACT-Europe mentioned.

* Ada 2005 features in GCC 3.4.

Policy for libraries:

* Turn the chapter into one node per section.

* New section: "Building libraries for reuse".

* New section: "Mandatory files provided by the -dev package", contains the former sections about source, object, *.ali, and library files, now in subsections.

* Mention recently added support for *.ali files in lintian, linda and dh_fixperms.

* Replace @code{} with @file{} in many places.

* Section "Shared libraries" renamed to "Files provided by the shared library package".

The Debian Policy for Ada is on my web site in HTML [1] and Texinfo [2] formats. In addition, I intend to upload it to Debian if my sponsor agrees. This means that it will be available a Debian package named debian-ada-policy. This package will include the document in Info, ASCII and PDF formats in addition to HTML and Texinfo source.

[1] http://users.skynet.be/ludovic.brenta/debian-ada-policy.html

[2] http://users.skynet.be/ludovic.brenta/debian-ada-policy.texi

[See also same topic in AUJ 25-5 (Jun 2004) p.64. --su]

GNAT 3.15p NPTL Support

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: Sun, 25 Jul 2004 22:40:31 +0200
Subject: GNAT 3.15p now supports NPTL on Debian
Newsgroups: comp.lang.ada

Today, gnat 3.15p-10 arrived in Debian Unstable. In this upload, I have back-ported several fixes from GCC's main line of development. One of these fixes allows programmers to use Ada tasking with the New POSIX Thread Library (NPTL) as provided by Linux 2.6 and glibc 2.3. It is thus no longer necessary to use "LD_ASSUME_KERNEL=2.4.1" as before.

I do not normally announce each new upload, but I thought that this particular fix would be of interest to more people. In particular, those of you not using Debian may want to extract my patches from Debian and apply them on your platform.

As usual, if no release-critical bug is reported against this package, it will reach Testing in 10 days.

It has been one year since I started to work on Debian, so you can call this the "anniversary edition" :)

Here is the complete changelog, for the curious: [See distribution --su]

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: Mon, 26 Jul 2004 00:17:09 +0200
Subject: Re: GNAT 3.15p now supports NPTL on Debian
Newsgroups: comp.lang.ada

Brian May wrote:

> What benefits does NPTL provide for Ada programs?

The benefits are not specific to Ada, but they are real. If you are building a web or application server based on AWS, or any kind of multithreaded server, you will be interested in NPTL. For an overview, see:

http://linuxdevices.com/articles/AT6753699732.html

Patched version of GPS in Debian

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: 22 Jun 2004 14:39:41 GMT
Subject: Re: GNAT Programming System Problems
Newsgroups: comp.lang.ada

Robert Love wrote:

> Ah, if it were only that easy. I work on a project where we have a contract with ACT so I can get the latest tools but the powers that be will not step up to the newer compilers. We're over 3 years behind and there is no concrete plan to modernize.

For Debian GNU/Linux, I have patched GPS so it does not generate the offending lines in project files. My patches also make it possible to compile GPS with GNAT 3.15p. You may want to apply them and rebuild GPS from source. On the URL below, look for the .diff.gz file, it contains the patches.

http://packages.qa.debian.org/g/gnat-gps.html [...]

Ada and Microsoft

Importing MS Excel files

From: Tanker <Tanker93@hotmail.com>
Date: Mon, 14 Jun 2004 18:32:47 +0200
Subject: import Excel file
Newsgroups: comp.lang.ada

I want to import data from an MS Excel file in my program. Can someone give me a hint how I can realize that. I already tried to read the exel file as a text_io and sequential_Io file but it's not possible as Text_IO 'cause it must be a sequential_IO (or direct_IO) file. The problem with the sequential_IO is, that I can't figure out how the form and substance of the file looks like (is it a record, what kind of record (contents)?)

From: Jeff C <jcreem@yahoo.com>
Date: Tue, 15 Jun 2004 01:11:20 GMT
Subject: Re: import Excel file
Newsgroups: comp.lang.ada

Tanker wrote:

> That was my first idea too, but the guy who I'm writing the program for, thinks that is a good thing to import an excel file directly without the step of making an csv file. A friend told me today, that there must be a ms tool for C++ (Ms Visual C++) that can sort out the relevant information from the excel file. I could bind the C++ code in my Ada program and use the Excel info.
> If it's too difficult to do, I will convince him that it must work with the cvs transformation.

There is already a tool that does it quite well. It is called Excel :)

Seriously, Excel (at least older version) could be accessed via COM libraries. You can auto create bindings to the Excel COM libraries pretty easily using GNATCOM

http://www.adapower.com/gnatcom/index.html

For some things it is really easy to use the COM interface. For others it’s not so easy.

Not sure where Excel falls.

From: Pascal Obry <pascal@obry.org>
Date: 15 Jun 2004 23:24:20 +0200
Subject: Re: import Excel file
Newsgroups: comp.lang.ada

Jeff C wrote:

> Seriously, Excel (at least older version) could be accessed via COM libraries. You can auto create bindings to the excel COM libraries pretty easily using GNATCOM

And via ODBC too.

Setting the Program Icon

From: Jano
Date: Mon, 14 Jun 2004 19:18:56 +0200
Subject: Re: set program icon
Newsgroups: comp.lang.ada

Tanker wrote:

> I want to set up a program icon for my program, but I have no idea how to do that. I'm using Adagide and the Gnat compiler 3.13p.

I've done it with 3.15p, I don't know if it works with previous versions.

Install gnatwin from ACT-Europe, there is a windows resources compiler (rcl I think). Create a .rc file (you can use visual studio or do it by hand, google should provide enough info) pointing to your icon. Compile with rcl the resource file and you'll get a .o object file. Include that file in the compiler chain to get it linked with your executable and voilà!

From: Tanker <Tanker93@hotmail.com>
Date: Mon, 14 Jun 2004 20:17:19 +0200
Subject: Re: set program icon
Newsgroups: comp.lang.ada

It worked perfectly: the rcl compiler created the *.o file from the *.rc file. Now I still got a problem: How can I include that file in the compiler chain in order to get it linked with my exe file?

From: Tom Moran <tmoran@acm.org>
Date: Mon, 14 Jun 2004 18:18:06 GMT
Subject: Re: set program icon
Newsgroups: comp.lang.ada

From an earlier article:

>Subject: Announce: .rc files with Gnat
Date: 2002-04-14 20:06:04 PST
David Botton tells me www.adapower.com/os/wglink.html now contains a linker shell I wrote to automatically handle Windows .rc files with Gnat. It should make life a little easier for both newbies and professionals using Gnat with MS Windows.

From: Bernd Specht
<Bernd.Specht@gmx.com>
Date: 15 Jun 2004 19:34:34 GMT
Subject: Re: set programm icon
Newsgroups: comp.lang.ada

Tanker wrote:

> Thanks Jano, it almost worked perfectly! But the program icon in the upper left corner of the program window is still the standard icon and not my program icon (shown in the Explorer!) Maybe someone can tell me the parameter for the gnat compiler?

It works for me since GNAT 3.12p. I don't use RCL, I use RC and CVTRES which comes (for example) with Aonix (the special edition included in Barnes book).[...]

References to Publications

Ada 2005 Presentation Update

From: Dirk Craeynest
<dirk@heli.cs.kuleuven.ac.be>
Date: 23 Jun 2004 23:49:07 +0200
Organization: Ada-Belgium, c/o Dept. of Computer Science, K.U.Leuven
Subject: Ada 2005 presentation updated on Ada-Belgium web site
Newsgroups: comp.lang.ada, fr.comp.lang.ada, be.comp.programming, nl.comp.programmeren

At the 9th International Conference on Reliable Software Technologies - Ada-Europe'2004 - held last week in Spain, Pascal Leroy, chairman of the ISO Ada Rapporteur Group (ARG), presented an updated version of the technical overview he gave at the Ada-Belgium event earlier this year, describing the most important improvements that are currently under consideration for inclusion in Ada 2005.

We are pleased to announce that the on-line version on the Ada-Belgium web pages has been updated using this recent presentation.

Check out "What's new on the Ada-Belgium web-pages?" at URL <http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/whatsnew.html> if you're interested.

[See also "Ada 2005 Presentation" in AUJ 25.1 (Mar 2004), pp.5-6 -- su]

Ada-Europe 2004 Papers Available On-line

From: Rod Chapman
<rod.chapman@praxis-cs.co.uk>
Date: 30 Jun 2004 07:46:23 -0700
Subject: New technical paper from Ada Europe 2004 now available
Newsgroups: comp.lang.ada

I'm pleased to announce the availability of two new technical papers that were recently presented at Ada Europe 2004.

Both are available now in PDF at www.sparkada.com

Note that the first of these jointly took the best paper award at the conference. The latter was awarded best presentation.

"High-Integrity Ada in a UML and C World", Peter Amey and Neil White

Abstract

The dictates of fashion and the desire to use "hot" technology not only affects software developers but also influences potential customers. Where once a client was just content to accept something that worked (actually, would be delighted to have something that worked) now they are concerned about the means by which it was constructed; not just in the sense of was it well-enough constructed but in the more malign sense of was fashionable technology used. This paper shows how the customer's desire to use de facto standards such as UML and their wish to use language such as C? perhaps to support a small or unusual processor; to integrate with other subsystems; for the perceived comfort of future portability; or for other, non-technical reasons? can be aligned with the professional engineer's need to use those tools and languages which are truly appropriate for rigorous software development.

"High-Integrity Interfacing to Programmable Logic with Ada"

Adrian J Hilton, Praxis Critical Systems Limited and Jon G Hall, Open University.

Abstract

Programmable Logic Devices (PLDs) are now common components of safety-critical systems, and are increasingly used for safety-related or safety-critical functionality. Recent safety standards demand similar rigour in PLD specification, design and verification to that in critical software design. Existing PLD development tools and techniques are inadequate for the higher integrity levels. In this paper we examine the use of Ada as a design language for PLDs. We analyse earlier work on Ada-to-HDL compilation and identify where it could be improved. We show how program fragments written in the SPARK Ada subset can be efficiently and rigorously translated into PLD programs, and how a SPARK Ada program can be effectively interfaced to a PLD program. The techniques discussed are then applied to a substantial case study and some preliminary conclusions are drawn from the results.

SPARK mentions at www.embedded.com & www.slashdot.org

From: Martin Dowie
<martin.dowie@btopenworld.com>
Date: 3 May 2004 00:11:10 -0700
Subject: SPARK @ embedded.com
Newsgroups: comp.lang.ada

SPARK (and Ada) get a nice mention in Jack Ganssle's column @

[http://www.embedded.com/showArticle.jhtml?articleID=19205573 --su]

From: Mark Lorenzen <mark.lorenzen@ofir.dk>
Date: Wed, 19 May 2004 22:27:14 +0200
Subject: Jack Ganssle's review of the SPARK book is mentioned on /.
Newsgroups: comp.lang.ada

The hacker site /. has an article referring to a review of the SPARK book that has previously been mentioned in this group.

http://books.slashdot.org/article.pl?sid=04/05/19/190235

Not surprisingly, a lot of negative feedback is written in the discussion forum, including critic of Ada, SPARK and who knows what. It is actually pretty irritating to read the idiotic comments on software engineering (or rather hacking/coding) and on why it would be much better just to test the program.

Maybe someone with the time and energy should reply to some of the postings - but it is probably just a waste of time.

From: Berend de Boer <berend@xsol.com>
Date: Wed, 26 May 2004 08:25:09 +1200
Subject: Re: Jack Ganssle's review of the SPARK book is mentioned on /.
Newsgroups: comp.lang.ada

Typical comment:

"I don't know a lot about this stuff, but ...."

Modded 5, Insightful of course.

Don't waste your time.

DDC-I Online News

From: jc <jcdk@ddci.com>
Date: Tue, 8 Jun 2004 15:45:05
Organization: DDC-I
Subject: Real-Time Industry Updates - News from DDC-I
To: 5D June 2004 Online News DK <jcdk@ddci.com>

DDC-I Online News

Real-Time Industry Updates - News from DDC-I

June 2004, Volume 5, Number 6 - [http://www.ddci.com/news_vol5num6.shtml]

A monthly news update dedicated to DDC-I customers & registered subscribers.

This Month:

* SCORE-653 Taking Off With Developers

Several Programs Currently Evaluating

* DDC-I's SCORE IDE Offers JTAG

Integrated support for the industry leading Abatron BDI2000 JTAG

* Thoughts from Thorkil - Number Conversions in DACS-80x86

Hexadecimal Notations in Embedded Applications

* Welcome to the Wide Open Spaces

An Intriguing Approach to Group Interaction

From: jc <jcdk@ddci.com>
Date: Thu, 1 Jul 2004 14:43:57
Organization: DDC-I
Subject: Real-Time Industry Updates - News from DDC-I
To: 6D July 2004 Online News DK <jcdk@ddci.com>

DDC-I Online News

Real-Time Industry Updates - News from DDC-I

July 2004, Volume 5, Number 7 - [http://www.ddci.com/news_vol5num7.shtml]

A monthly news update dedicated to DDC-I customers & registered subscribers.

This Month:

* DDC-I's Multi-Language SCORE® IDE Adds FORTRAN

Software Developers Maintaining Legacy Code SCORE a Major Victory

* DDC-I Announces New Distributor

New Champion in the French Real-time Development Community

* Thoughts from Thorkil

Date, Time and Ada (2)

* What the #$*! Do We Know

It's a story; it's a documentary. It's humorous but thought provoking.

Ada Inside

Ada Usage on the Boeing 7E7

From: Ed Falis <falis@verizon.net>
Date: Wed, 26 May 2004 01:28:40 GMT
Subject: Re: Boeing 7e7 and Ada?
Newsgroups: comp.lang.ada

Richard Riehle wrote:

> I asked this question once before, but received no authoritative answers. Does anyone know, for certain, whether Ada is being used for the software on the Boeing 7e7?

At least in part, yes. Wind River's Tornado AE653 (an implementation of ARINC 653) is slated to be used for the IMA architecture. GNAT is slated for use on at least some of the subsystems. I do not know the extent to which Ada is required, recommended or encouraged.

[See also "Wind River Teams with AdaCore on Platform Safety Critical ARINC 653 for Use in Boeing 7E7 Common Core System" in AUJ 25-2 (Jun 2004) p.59. --su]

Indirect Information on Ada Usage

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

From: Amy Wong <amy.wong@windriver.com>
Date: 14 Jul 2004 15:32:10 -0700
Subject: Ada Software Engineering Instructor Position
Newsgroups: comp.lang.ada

Wind River Systems in Alameda, CA is looking for an Ada Software Engineering Instructor. This person can be based in Alameda, CA but we are open to other locations across the U.S. as the position will require 50% travel.

Responsibilities include presentation of technical training courses to customers working with computerized tools for the design of embedded systems in a variety of application areas.

Primarily responsible for teaching courses on Wind River's safety critical products in Ada. Other course areas include VxWorks and the Workbench/Tornado family of development tools, device driver writing, BSP development, networking, and Wind River Platform components.

The position requires an intelligent, self-motivated individual who can work independently, be responsible for deadlines, as well as perform effectively in a team-oriented training department. Excellent communication skills both verbally and in writing, being able to create exemplary Ada-code or pseudo code models, and being able to learn new technologies quickly are the fundamental requirements for this position.

Major duties:

- Teach Ada portion of training courses

- Teach VxWorks AE653, Workbench/Tornado, and other workshops to Wind River Systems customers and employees

- Develop new course material and maintain existing course material

- Perform other training department tasks as needed (learn new technologies, develop and test lab examples etc.)

- 50% travel required within North America

Requirements:

- Ada programming experience and being able to explain the code

- Previous experience with VxWorks and Tornado highly desired

- Strong interpersonal and written communication skills

- Experience programming in real-time or multitasking environment (VxWorks, UNIX, Linux, etc.)

- A desire to teach highly technical material to experienced computer professional

- Strong problem solving skills in short time (on the spot)

Desired skills:

- Commercial technical training experience

- Familiarity with ARINC 653

- Familiarity with DO-178B or similar standards

- Familiarity with Ada Core Technologies GNAT tools

- Embedded real-time programming experience (VxWorks, pSOS)

- Experience with standard bus hardware (VME, PCI) and/or embedded software including device drivers

- Experience developing commercial training curriculum (MS PowerPoint, MS Word, Adobe Acrobat etc.)

Preferred experience:

- BS in Computer Science or Electrical Engineering or 5 years of equivalent industry experience

- Advanced degree in Computer Science or Engineering is a plus

URL: http://job.monster.be/ […]
Date: Thu, 29 Apr 2004 13:09:25
Subject: BE-Brussel-Brussels-Ada developer

[...] With a team of 240 persons, Aubay BeLux offers IT-services either as time & means consultancy or as fixed-priced projects. The service lines offered by Aubay BeLux are:

- Information Systems Development based on open platforms, techniques and methods (J2EE, UML, SOAP, Application Servers, .NET, Relational Databases, etc.)

- Application & Infrastructure Support (system management [UNIX or Windows], network management, application maintenance, etc.)

- Implementation of Document, Content and Knowledge Management Systems (documentary databases, classification tools, conversion tools, portal building technologies, XML/SGML, Business Intelligence tools, etc.)

- Real-time Systems Development (specialized languages, real time kernel, etc.)

- Implementation of Quality Assurance Processes (Quality Assurance & Control)

- Audits and Technical Architecture Studies

For more information visit our website: www.aubay.be

We are current looking for a Ada developer:

Expertise of several years with Ada is mandatory.

Experience in ATC is an advantage.

Higher education in Computer Science or engineering.

Very good knowledge of Dutch or French and English [...]

URL: http://job.monster.be/[…]
Date: Thu, 29 Apr 2004 13:09:46
Subject: BE-Brussel-Ada SOFTWARE ENGINEERS

Thales Information Systems is one of the most structured European IT Services players in the range of 5000 persons and 500M Euros with main positions in France, Belgium, Germany, UK, Switzerland and Spain.

Expertise and reputation have enabled the group to gain a broad customer base including major multinational groups, and further develop its activities in a large og high growth markets and demanding business activities:

- Industry (Manufacturing, Service, ...)

- Finance (Bank, Insurance, Financial and other Financial Services)

- Utilities (Energy, Telecom, Transportation)

- Public Sector (Administration, Defence, Health)

Throughout these market segments, Thales IS is delivering 3 core offers:

- Integration of Management Systems (including ERP, CRM, PDM, SCM, EAI)

- Integration of Technical Systems (Supervision and control systems)

- IT Outsourcing: Management Services (infrastructure outsourcing), Third Party Maintenance and Application Management.

[...] You will be integrated in a strategic project including architectural & detailed design, development of the application, programming, testing and writing of the design documentation.

Profile:

Education: Civil Engineer, Industrial Engineer, Graduate in Computer

Sciences or similar experience

Knowledge of a programming language preferably Ada 83-95 or C++

Developer & analyst designer

Team spirit and good methodology [...]

URL: http://job.monster.be/[…]
Date: Fri, 6 Aug 2004 11:46:08
Subject: BE-Brussels-Belgium-Ada Developer

A client of Huxley Associates based in Belgium urgently requires an Ada 95 Developer with at least 3 years development experience. You must have had recent exposure to Ada development. Knowledge of programming on UNIX and C++ would be desirable but not essential. This role will involve elements of design and development. Candidates must be English speaking where French would be a plus. Ada 95 DEVELOPER; UNIX; C++

URL:
http://job.monster.be/getjob.asp?JobID=23075858
Date: Fri, 6 Aug 2004 11:46:43
Subject: BE-Vlaams-Brabant-Ada developer

[...] We are currently looking for an Ada developer:

* Several years of experience in a similar function using Ada 95/Linux

* Experienced in Air Traffic Management (ATM) is a major advantage.

* Full university level, or graduate in Computer Science or engineering

* Fluent in French, Dutch and English [...]

Ada in Context

Unicode Support in Ada

From: Brian Catlin <BrianC@sannas.org>
Date: Tue, 11 May 2004 17:45:41 GMT
Subject: Supporting full Unicode
Newsgroups: comp.lang.ada

The complete definition of Unicode allows for 2-,3-, and 4-byte characters. How is this supported in Ada95 and Ada0y?

From: Martin Krischik <krischik@users.sourceforge.net>
Date: Wed, 12 May 2004 11:30:26 +0200
Subject: Re: Supporting full Unicode
Newsgroups: comp.lang.ada

Built in for Ada95 is ISO-8859-1 and ISO-10646.

Currently XML/Ada has full support for Unicode.

BTW: is 1, 2, 4-byte characters.

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: 12 May 2004 07:44:56 GMT
Subject: Re: Supporting full Unicode
Newsgroups: comp.lang.ada

I am not aware of any differences between Ada 95 and Ada 2005 in that respect. Ada 95 has a type Wide_Character, "whose values correspond to the 65536 code positions of the ISO 10646 Basic Multilingual Plane (BMP)." (RM 3.5.2(3)). So, Ada 95 supports the UCS-2 encoding natively. The other standard type, Character, is defined as Latin-1.

As you can see, there is no standard support for 3- and 4-byte characters; you

would have to support them in a nonstandard way, e.g.

```
type Wide_Wide_Character is
   mod 2**32; -- UCS-4
type Wide_Wide_String is
   array (Natural range <>) of
      Wide_Wide_Character;
```

But I would favour using UTF-8 as the internal encoding anyway. It is easy to define a UTF8_String type similar to the above. GtkAda has such a type, as GTK+ uses UTF-8 as both internal and external encoding.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 12 May 2004 20:15:14 -0500
Subject: Re: Supporting full Unicode
Newsgroups: comp.lang.ada

Brian Catlin wrote:

> The complete definition of Unicode allows for 2-,3-, and 4-byte characters. How is this supported in Ada95 and Ada0y?

It hasn't completely been decided yet. AI-285 is the AI for that. There are some issues that have to be taken up at the WG9 level (the ARG cannot make a decision because some countries oppose the only practical solutions).

It doesn't look like there is going to be any real support for UCS-8 or UCS-16, though.

From: Marius Amado Alves <amado.alves@netcabo.pt>
Date: Thu, 13 May 2004 11:09:14 +0100
Subject: [OT:fun] Re: Supporting full Unicode
Newsgroups: comp.lang.ada

[When the extraterrestrials come...]

Unicode will be renamed Earthcode, and become a part of Galacticode, a 42-bit code space managed from Vulcan, which will eventually be renamed to Lacteacode and become a part of Clustercode. By then language is not evolving anymore so there is a code for each word. Clustercode will finally become a part of Unicode--this time named right. By then everything has been said so there is a code for each sentence. At some point in this timeline the ISO (Intergalactical Standards Organization) went out of business because 1,000,000,000,000-page printed volumes became unpractical to ship.

Porting Ada Source Code & GNATPrep

From: "Giacomo Polizzi"
Date: Mon, 19 Jul 2004 09:42:27 +0200
Subject: Porting Ada source
Newsgroups: comp.lang.ada

I have to port an Ada program from a UNIX DEC Alpha machine with a DEC Ada compiler to a Linux pc machine with gnat (gcc) compiler.

The task is to have, if possible, a unique source code compilable on both platforms but there are the following problems:

1) some system packages that execute the same kind of operations have different names in the two compilers

2) some system functions (for example mathematical function) have different names in the two compilers

3) the word length is 64 bits on DEC Alpha and 32 bits on pc so it is not always possible to use the same standard type (for example the long type is 64 bits on Alpha and 32 bits on pc)

4) the standard type (long long) used by gcc on pc to solve the previous problem is not supported by the DEC Ada compiler

5) the different word length modifies some structure length used to define interface messages with other external programs

Is there something like #IFDEF of C language that I can use in AD?

Is there any other kind of solution for the above problems?

From: "Martin Dowie" <martin.dowie@btopenworld.com>
Date: Mon, 19 Jul 2004 18:05:58)
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

You could (possibly) keep the original package spec and then select either the original package body or a new 'wrapper' package body at build time.

[...] Don't use the standard types! They are a convience for writing [quick & dirty] programs and should never be used in production code.

You could look at "Adasubst" at http://www.adalog.fr/compo2.htm#Semtools

This can look through all your code and replace the offending "Long_Long_Float" with "My_Float" (or whatever you define).

From: Marin David Condic <mcondic@acm.org>
Date: Mon, 19 Jul 2004 12:14:15 GMT
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

Part of the problem seems to be that the OP is working with existing code that nobody apparently thought needed to be portable. Trying to dramatically restructure the code at this stage starts becoming problematic. Portability can be achieved, but only if you work really hard at it from the start. Of course there might be easier fixes to the problem now if Ada had some kind of conditional compilation directive - but since those are viewed as "Morally Evil" I guess people stuck with practical problems should just go use a different language. :-)

From: Wes Groleau <groleau+news@freeshell.org>
Date: Mon, 19 Jul 2004 14:33:31 -0500
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

Martin Dowie wrote:

> Don't use the standard types! They are a convience for writing q&d programs and should never be used in production code.

Advice that one could easily take too far. I know of a major project where all sorts of unnecessary code had to be written for handling text because String and Integer were forbidden types!

From: Wes Groleau <groleau+news@freeshell.org>
Date: Mon, 19 Jul 2004 15:29:13 -0500
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

Larry Kilgallen wrote:

> What is the advantage of forbidding "string"?

None that I know of. But the powers that be when the project started decreed that all predefined types are non-portable, therefore all types must be subtyped or derived from types created AND rep-spec'd for the project. Much of Standard and System was re-implemented to support these types. Character was the only predefined type exempt (why was that different?), so arrays of Character with "portable" index types were used instead of String.

Assigning string literals to these arrays was either humorous or nerve-wracking, depending on how much time pressure you were under.

Interestingly, string literals and character literals were one of the Ada 95 conversion headaches. :-)

From: "Richard Riehle" <adaworks@earthlink.net>
Date: Sat, 31 Jul 2004 16:04:35 GMT
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

Porting from DEC to anything else can be very entertaining, especially when numerics are involved. The sign bit, on the VAX, is not located where you might think.

Confronted with this problem, in a one-time conversion, many years ago, we opted to convert the VAX numerics to human-readable (using Text_IO) and read them back into the target system, again using Text_IO. This is a brute-force approach, but it was suitable to the problem at-hand at that time.

NOTE: For some reason, many long-term users of Ada do not realize that these conversions are already a part of the language and found in the nested IO packages within Text_IO.

We first tried to create an algorithm to convert the VAX word into the word of our target platform. That turned out to be

an ugly mess with lots of code to verify the correctness of each translated word. It is even more fun when the DEC word represents a floating point value and needs to be converted to a floating point value on the target.

I realize this is only part of your problem, but others may want to contribute additional recommedations based on their direct experience.

From: Jeffrey Carter <spam@spam.com>
Date: Tue, 20 Jul 2004 18:49:51 GMT
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

Maybe. But the mission is probably to make the SW portable, so a certain amount of redesign is probably acceptable.

From: "Randy Brukardt" <randy@rrsoftware.com>
Date: Tue, 20 Jul 2004 17:51:57 -0500
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

If it is a long-lived system (and this one appears to be) a quick and dirty solution simply sets the developers up for many years of pain in maintenance. It's better to do it right the first time (or, as in this case - a very common one, the second time).

Case in point: when we ported Janus/Ada to the U2200 computers, we had to change a lot of assumptions about the sizes of things. (We had of course assumed character = byte = 8 bits, and of the U2200 was a 36-bit word machine.) Rather than simply coding a different set of assumptions, we decided to restructure everything so that the compiler depended on a small set of packages that encoded host and target information. That eliminated future redos for similar issues. I'm sure the customer would have been happier in the short run if we hadn't restructured everything, but it meant that virtually all bug fixes only needed to be made once -- a huge time savings in the long run.

From: Marin David Condic <mcondic@acm.org>
Date: Tue, 20 Jul 2004 11:59:50 GMT
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

Right. The only problem being that you are suddenly into rather significantly changing the software. You're repackaging things and restructuring things and you've got to try to set up some kind of configuration control process to accommodate two builds. If you are starting from scratch and you know you have to maintain portability this might be a valid approach (though I've seen this get waaaaayyy overblown too!)

But the mission is to take code body X and get it from its existing platform onto another - not redesign the system. Time spent reorganizing the code and setting up a build process does not directly move that mission forward. Hence, it would be nice to get some kind of quick and dirty fix that doesn't mean spending excessive time reworking the existing product. A conditional compilation directive can be that sort of fix. It's just that it is viewed dimly as a sub-optimal formal technique. Oh well....

From: "Robert I. Eachus" <rieachus@comcast.net>
Date: Tue, 20 Jul 2004 20:35:07 -0400
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

[...] Quick and dirty gets to "sort of working" faster, but doing it right seems to get to working faster, and definitely gets to no bugs much faster.

However, there is an issue here that may require living with two versions for a long-time: the need to compile using the DEC Ada compiler which AFAIK is Ada 83 only. I believe that GNAT is available for some Alpha OSes. (I don't know which UNIX variant this is.)

I think that the right path forward would be to migrate the code to GNAT, then use the GNAT version on the Alpha system. It may be a lot simpler than any other approach, and it certainly should reduce the support costs going forward.

From: Peter Amey <peter.amey@praxis-cs.co.uk>
Date: Tue, 20 Jul 2004 14:13:52 +0100
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

Giacomo Polizzi wrote:

> I have to port an Ada program from an UNIX DEC Alpha machine with a DEC Ada compiler to a Linux PC machine with GNAT (GCC) compiler.

Like others have said, the problem is best avoided by design for portability. However, that doesn't help if it wasn't!

I haven't seen a mention of the rather excellent GnatPrep which might just help:

http://gcc.gnu.org/onlinedocs/gnat_ugn_unw/Preprocessing-Using-gnatprep.html

From: Volkert <volkert@nivoba.de>
Date: 20 Jul 2004 23:20:22 -0700
Subject: Re: Porting Ada source
Newsgroups: comp.lang.ada

For moving from Ada 83 to Ada 95 read the Ada Compatibility Guide:

http://unicoi.kennesaw.edu/ase/ase02_01/docs/compat/compat-guide6-0.pdf

We are still migrating more than 3 mio loc of Ada 83 code (OpenVMS, DecAda) to GNAT (OpenVMS). For such a large application it is really hard work. The Source has to work for some time on both compilers. The interesting point is, that we have found to all "source-code" problems solutions without using tools like gnatprep.

If it is a valuable piece of software, I would expect that you have a support contract with ACT. They will assist you.

Memory Management and Productivity

From: Russ <18k11tm001@sneakemail.com>
Date: 16 Jun 2004 21:56:23 -0700
Subject: memory management and productivity
Newsgroups: comp.lang.ada

The following is an excerpt from an article featured on Slashdot today [2004-06-16 --su]. The excerpted paragraph is not even the main topic of the article, but it made wonder what chance Ada has in the future without automatic memory management. It seems to me that garbage collection should be optional within the language.

Excerpt from http://www.joelonsoftware.com/articles/APIWar.html

How Microsoft Lost the API War

By Joel Spolsky

....

"A lot of us thought in the 1990s that the big battle would be between procedural and object oriented programming, and we thought that object oriented programming would provide a big boost in programmer productivity. I thought that, too. Some people still think that. It turns out we were wrong. Object oriented programming is handy dandy, but it's not really the productivity booster that was promised. The real significant productivity advance we've had in programming has been from languages which manage memory for you automatically. It can be with reference counting or garbage collection; it can be Java, Lisp, Visual Basic (even 1.0), Smalltalk, or any of a number of scripting languages. If your programming language allows you to grab a chunk of memory without thinking about how it's going to be released when you're done with it, you're using a managed-memory language, and you are going to be much more efficient than someone using a language in which you have to explicitly manage memory. Whenever you hear someone bragging about how productive their language is, they're probably getting most of that productivity from the automated memory management, even if they misattribute it."

From: Martin Dowie <martin.dowie@baesystems.com>
Date: Thu, 17 Jun 2004 09:07:56 +0100
Organization: BAE SYSTEMS
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

You will be delighted to hear that Ada already supports [Garbage Collection

(GC)] through the use of Controlled types then. And if that weren't enough you can grab "Boehm-Demers-Weiser" GC from AdaCL @ http://adacl.sourceforge.net/.

[See also "Does Ada Need Garbage Collection?" in AUJ 22-3 (Sep 2001) -- su]

From: Tom Moran <tmoran@acm.org>
Date: Thu, 17 Jun 2004 05:15:42 GMT
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

I guess Ada is a managed-memory language then, since I normally grab a chunk of memory without thinking about how it's going to be released. In fact I don't even think about whether the compiler might actually be doing a heap allocation with hidden pointer, or doing the stack allocation that appears to be happening. And when I use a Controlled type from some library, I let its Finalize do any memory worrying that's needed. Only on rare occasions do I explicitly do a "new" and have to remember to do a "Free".

But I sincerely doubt anyway that memory management is the proverbial silver bullet. Yes, it's faster, as well as less buggy, to program in Ada, but it's still not a snap.

From: Martin Dowie <martin.dowie@baesystems.com>
Date: Mon, 21 Jun 2004 08:46:36 +0100
Organization: BAE SYSTEMS
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

Russ wrote:

> Well, if initialize and finalize are fundamentally no different than constructors and destructors, then I conclude that Ada does not have automated memory management any more than C++ has it. (I may be a bit slow, but I catch on eventually.)
> As for using a library that has memory management built in, that sounds great. I'd just feel a bit better about it if it were in a standard library that everyone agrees on and comes with the language, like STL in C++. Then you could honestly say that Ada has automated memory management built in. (And it would be better than Java's because it would be optional.)

Ada does not forbid an implementation for performing GC but there has been zero customer call for it. That's no too surprising - "Sorry, you can't apply brakes/launch missile/shutdown reactor as I'm busy garbage collecting." would not go done well.

But for desktop apps or even business apps, it might well be appropriate.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 21 Jun 2004 10:23:11 +0200
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

The difference is that in Ada there is much lesser need in objects allocated in the heap. That makes things a lot easier and safer. Moreover it is also much more efficient, especially in a parallel environment.

From: David Starner <dvdeug@email.ro>
Date: Tue, 22 Jun 2004 02:39:35 GMT
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

Russ wrote:

> Are you saying that an Ada compiler can implement GC or not? That seems a bit inconsistent to me. How can the same language have it both ways?

Every language has it both ways. You could link the Boehm-Weiser garbage collector into a C program without any change to the source outside a couple defines on the command line. A Lisp interpreter could choose not to garbage collect memory. Like interpretation, GC is more a feature of choices and tradition than an enforced standard.

The advantage of Ada is that the strong typing makes it possible for an advanced GC to find pointers and adjust them when moving things in memory. Ada also makes many of the small, temporary memory uses found in C unnecessary by allowing you to pass things by reference and to pass arrays.

> What if someone tries to port an Ada application that has its own memory management to an Ada compiler that already has it by default?

It depends on the code and on the GC system. If you depend on the details of memory management, then you depend on the details of memory management. If it's pure Ada, it should work; if it's not, it might not work. In any case, well-designed memory management systems are usually easy to pull out or at least write a dummy replacement for.

> Which is why it should be under control of the application developer, it seems to me.

But it is. The application developer chooses which compiler to use and which flags to compile it with.

If you're compiling for a LISP machine or a JVM, you probably don't have a choice whether or not to use GC. As the only Ada compilers with built-in GC right now are those which compile to the JVM, they don't offer a choice.

From: Robert I. Eachus <rieachus@comcast.net>
Date: Sat, 26 Jun 2004 12:00:20 -0400
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

Symbolics had an Ada compiler for their systems, which had global garbage collection. I'm not sure about the R1000 Rational hardware.

But in any case, this has been the history of global garbage collection in Ada. If the OS supports it, the compiler will do what is necessary to run in that environment. But it is seen more as an extra development cost than a desired feature. (I'm not saying that is why Symbolics went bankrupt. But if there was a big market for Ada with garbage collection, Symbolics might still be in business.)

From: Ole-Hjalmar Kristensen <ole-hjalmar.kristensen@sun.com>
Organization: Sun Microsystems
Subject: Re: memory management and productivity
Date: 22 Jun 2004 10:03:04 +0200
Newsgroups: comp.lang.ada

But the only language which I have used so far which got it quite right IMHO is Modula-3. The programmer is free to choose, and the language guarantees that GC is available.

From: Russ <18k11tm001@sneakemail.com>
Date: 22 Jun 2004 21:41:20 -0700
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

Bingo. That's exactly what Ada should have. And it shouldn't take a rocket scientist to figure that out.

From: Larry Kilgallen <Kilgallen@SpamCop.net>
Date: 23 Jun 2004 07:11:17 -0600
Organization: LJK Software
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

ACT says they have not really seen market demand from their customers, and some of them are presumably rocket scientists.

Absent customer demand, there seems no justification for a _requirement_ since Ada already has a _permission_.

From: Björn Persson <rombo.bjorn.persson@sverige.nu>
Date: Wed, 23 Jun 2004 17:21:59 GMT
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

Well, I'd think rocket scientists don't want garbage collection. Their programs are real-time systems.

Developers of desktop applications are more likely to want garbage collection, but very few of them use Ada, and of those who do, probably very few can afford support contracts from ACT.

From: Ole-Hjalmar Kristensen <ole-hjalmar.kristensen@sun.com>
Date: 24 Jun 2004 09:35:57 +0200
Organization: Sun Microsystems

Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

In that case, you do not want any dynamic allocation whatsoever, so the availability of garbage collection is not important. On the other hand, *IF* you can live with dynamic allocation, there are garbage collection algorithms which will give hard real-time performance, provided you have an upper bound on the size of your objects. Baker's treadmill, the incremental copying collector, and reference-counting schemes can all do this with a very slight loss of generality.

From: Russ <18k11tm001@sneakemail.com>
Date: 23 Jun 2004 11:18:09 -0700
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

The rocket scientists probably don't need or want automated memory management for their real-time applications, but desktop application developers probably do. Why not make them both happy?

Oh, the desktop developers are not "demanding" it? Of course they aren't, because they aren't using Ada in the first place. And why aren't they using Ada? Perhaps in part because it doesn't have GC.

Why wait for your current customers to "demand" something before adding a basic feature that could turn *potential* customers into *actual* customers? I'm afraid that such short-sighted views will only perpetuate the demise of Ada.

From: David Starner <dvdeug@email.ro>
Date: Wed, 23 Jun 2004 23:18:25 GMT
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

So instead they use ... C++. Ada is not going to make a dent in Perl, Python, or Lisp(s), because they aren't designed for the same types of jobs. As for Java, ACT produced a version of GNAT (JGNAT) that compiled to the JVM (and thus was GCed). They found no (few?) takers, and stopped supporting it. GC just ain't going to get a bunch of programmers using Ada.

From: Russ <18k11tm001@sneakemail.com>
Date: 23 Jun 2004 23:40:30 -0700
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

Larry Kilgallen wrote:

> I have the feeling you don't have experience in marketing or overall company finance.

No, I don't. But it seems to me that Ada has a golden opportunity here to provide a clear advantage over both C++ and Java -- and it is declining.

C++ has no garbage collection, and development time is two to three times greater as a result (not to mention memory leaks all over the place). Java has GC, but everyone is working overtime to make it work for real time. The obvious solution is to have GC available on demand -- but *optional* at the discretion of the developer.

You Ada enthusiasts tell me that the Ada standard "allows" GC but does not require it to be available. Well, as an engineer who would like to promote Ada, I honestly don't know what that means. Does Ada have standard automated memory management or doesn't it? Can I count on it, or can't I? If not, what good is it to me?

Maybe you think I am just too simple minded. Well, maybe I am. But so are a lot of other people, including many of the decision-making managers you guys are so fond of. When they compare languages (if they do), one thing they do is to set up a list of check boxes, and one is for GC. As a rule of thumb, a check beats a question mark every time.

From: Russ <18k11tm001@sneakemail.com>
Date: 25 Jun 2004 16:35:53 -0700
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

I don't know much about memory management or garbage collection, but I do know that it is considered a great benefit of Java and many other languages. Whatever Ada provides in this regard should be optional at the discretion of the *developer* (not the compiler vendor), and it should be at least as good as the GC in Java for those who choose to use it that way.

[...] I think you are missing a huge point here. The advantages of automatic memory management are that it is (1) convenient for the developer and (2) much more reliable, just as the automatic array bounds checking is more reliable than a developer's hand-rolled bounds checking.

You may neither want nor need it for your application, and you shouldn't be forced to use it, but the desktop and web developers should always have it available if they want it. They shouldn't be denied a valuable feature just because you don't want it. [...]

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Sat, 26 Jun 2004 11:27:07
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

Russ wrote:

> Look at the amazing popularity of Java. I would venture to say that it is due, in large part, to Java's garbage collection.

I don't believe for a minute that Java is popular "in large part" because of GC - much more to do with Sun's $$$$ and a huge built in _standard_library.

Having said that, I don't see any harm in making GC in Ada much more mainstream.

From: Ed Falis <falis@verizon.net>
Date: Sat, 26 Jun 2004 13:45:12 GMT
Subject: Re: memory management and productivity
Newsgroups: comp.lang.ada

Unfortunately, there are counter-arguments to this conjecture, anyway.

Consider Eiffel, a very nice language, quite Ada-like in many ways, with garbage collection and several innovative features (eg design by contract). It is not as popular or as widely used as Ada.

Consider that both Aonix (from Intermetrics/Averstar) and AdaCore provided Ada compilers for the JVM, including garbage collection and robust bindings to the Java class libraries. Neither was commercially successful.

Consider that when I was at Aonix, I built a setup allowing the use of the Great Circle garbage collector with ObjectAda, and that we advertized the availability of this capability. One copy of GC was sold with ObjectAda if I remember rightly.

So, IMO, garbage collection is a nice option to have, but I doubt that it's of the essence of Java's popularity.

By the way, I happen to enjoy programming in languages that provide GC, and was pretty heavily involved in almost all of the above.

Ada at the ACM

From: Robert C. Leif <rleif@rleif.com>
Date: Sun, 13 Jun 2004 19:36:38 -0700
Subject: ACM
Newsgroups: comp.lang.ada

When I renewed my ACM membership, I had to answer Question 7. Ada was NOT given as a selection. BASIC, dialects of C including two forms of Java, .Net, and XML were listed. I hope that the SIGAda leadership and others active in ACM alert this organization to Ada.

Otherwise, I would prefer to be affiliated with the IEEE. At least, it includes the word engineering in its name.

7. Which programming languages do you know/use, or plan to purchase within 12 months? (Check all that apply). No Ada!

Ada Usage in General Aviation (GA)

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: 10 May 2004 23:31:01 +0200
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

I happen to work in that particular market. Barco makes cockpit displays. The vast majority of the software is written in Ada; it is my understanding that Ada is still mandatory for DO-178B certification, but I may be wrong on this. Now the general aviation market is only a part of the markets that Barco addresses, and usually GA aircraft tend to be on the higher end of the spectrum (i.e. at least turboprop, not piston-engined).

From: Britt Snodgrass <britt@acm.org>
Date: 11 May 2004 07:29:33 -0700
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

DO-178B (see section 4.4.2) does not specify or recommend any particular software language, regardless of level. Some C language applications such as RTOS's have been certified to DO-178B level A. However the ARINC 653-1 standard (last updated in Oct 2003) does recommend Ada (see section 1.4.2).

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Tue, 11 May 2004 20:12:41
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

Martin Dowie wrote:

> DO-178B has never mandated Ada - but has 'highly recommended' it.

Sorry, it's Def-Stan-0055/56 that has the recommendations

From: Richard Riehle <adaworks@earthlink.net>
Date: Wed, 12 May 2004 00:07:53
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

Marin David Condic wrote:

> You could ask, but the answer is "No - but we'll let you pay to develop it and you can wait a year to get it." You're absolutely right on this score. *NOBODY* is going to pay a vendor and wait for the privilege of using Ada. They're going to select their board/chip and they're going to use whatever compilers/development environments are available for that hardware. If Ada isn't sitting on the shelf waiting to be bought, then Ada doesn't play.

Ada compiler publishers have been dwelling in the DoD world for such a long time that any tendency toward entrepreneurialism has vanished. I talked to one DoD contractor's management a few years ago and suggested they had an opportunity to enter a marketplace (not with Ada, but other skills) and become successful in that market. Their response, "But who is going to fund it?"

There is a risk associated with any new venture. Entrepreneurs tend to be driven by their vision, their dedication to that vision, and their unwavering confidence that it is a vision with commercial value. At one time, before the wimps took charge, Aonix seemed almost ready to pursue that kind of vision.

Among those who have grown up in the DoD contracting world, courage seems in short supply. Some of the people who build tools for Ada don't use their own Ada compilers and tools for building their other products. I don't need to identify those companies. They know who they are. However, they don't have a sense of how ashamed they should be of their lack of vision, lack of courage, and lack of entrepreneurial will.

"We will only build a product if someone asks for it," is not a particularly good business strategy. One can keep a company alive, for a while, using that strategy, but it is not a sustainable posture. Entrepreneurs are constantly trying to find new markets, not simply cling to existing ones.

There have been a few examples of risk takers in the Ada industry. RR Software comes to mind. Meridian comes to mind. OC Systems.

There are a few others. The fact that these have not been a resounding success has acted as a deterrent for others. One of the companies that had a chance to make Ada popular, Rational, flubbed that opportunity. I don't know if it was inept management, lack of courage, or just being too busy to put any energy into seeking commercial success for Ada.

Whatever it was, the language product that got them started, Ada, seems to have vanished from Rational's main marketing thrust.

The only companies that actively and energetically market their Ada products, at present, seem to be ACT and DDC-I. Every month, I get a newsletter from DDC-I that updates me on what they are doing, customer news, product news, and even a little column by a guest writer. No other Ada compiler publisher takes the trouble to do anything like that. If you want to be on their distribution, I suggest you send them an email. They are quite easy to deal with on such matters.

Marin is correct when he suggests that the only way to make Ada successful is for people to being creating products that use it. We can whine about the fact that more people are not choosing it, we can complain about the stupidity of the LM management on JSF, we can wring our hands about the downside of abrogating the mandate. None of that is worth much. What is worth a lot is for those who know and love Ada to build commercial products using it. Sell your shrink-wrapped Ada application to the general marketplace. Let people know you used Ada for development. Once we have those kind of successes, Ada can stand on its own and will be recognized for the value it actually provides.

From: Peter Amey <peter.amey@praxis-cs.co.uk>
Date: Wed, 12 May 2004 11:01:39 +0100
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

Bernd Specht wrote:

> Marin David Condic wrote:
> If Ada wants to be a player in General Aviation avionics it needs to do something to either a) target & capture some more "general" market that will provide the $$$ that will build the tools/talent/support
> There is a wide range of embedded applications on 8- and 16-bit processors like 8051 and HC11. But - no (Ada-) compiler vendor offers a compiler for such targets.

There might not be general Ada solutions for these processors but there a SPARK one. Because SPARK is designed to require little or no run-time library support it is possible to design and analyse in SPARK and then compile to C en route to running on small processors. The translation is simple because SPARK in unambiguous and because issues like array bounds violation have all been dealt with and eliminated at the SPARK design level. I am presenting a paper on this at Ada Europe this year.

From: Peter Amey <peter.amey@praxis-cs.co.uk>
Date: Thu, 13 May 2004 08:43:17 +0100
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

Marin David Condic wrote:

> The "Compile Ada to C then compile C to the target..." approach has been enormously popular and successful, hasn't it? :-)

Yes, considering we only came up with the SPARK -> C idea last year it has been rather popular :-)

> Nobody wants to use two compilers, nobody wants to cobble the whole mess together and nobody wants to fight all the rest of the development tools that will be targeted to C. The whole argument amounts to "Go out and spend lots of extra money, take lots of extra time and go through lots of extra pain in order to have the *privilege* of programming your little board in Ada - all for a job that might only involve a couple of guys for a few months." It has not sold well and I doubt it ever will. Ada is either right there on the shelf with an embedded development kit or the embedded guys go elsewhere.

There is some truth in what you say; the process has to be convenient. I think this hinders use of the approach for Ada in

general. The use of SPARK changes the gearing somewhat:

1. You are achieving more than "the privilege of programming your little board in Ada", you are gaining the ability to do very high levels of validation much earlier in the development process. Before you even get to C you can have eliminated use of unset variables, proved freedom from all run-time errors, shown that values remain in design-domain-defined ranges (regardless of what C predefined type they end up being implemented as), demonstrated separation of critical and non-critical information flows and even proved that certain important properties of the code hold.

2. The translation process can be completely transparent. As far as we are concerned on one project, we have what amounts to a SPARK to target compiler. We don't need to process the C manually at all.

3. If you are constrained to deliver something like MISRA-C then it is orders of magnitude easier to arrange for the translator to represent the exact SPARK design in something which is unequivocally MISRA-C than to demonstrate retrospectively that hand-crafted C conforms to the subset rules.

4. For very critical applications, the C provides an intermediate representation between the SPARK and the object code and this makes object code verification much easier than it otherwise would be. For example, array assignments at the Ada level are "unwrapped" at the C level. This intermediate representation allows one very hard step: comparing Ada to machine code to be replaced by two much easier ones: Ada to (simple) C and C to machine code.

5. You don't need, and therefore don't have to validate, a run-time library.

So far from having to "spend lots of extra money, take lots of extra time and go through lots of extra pain" we find the combination allows us to produce higher quality at lower cost than writing directly in C.

From: Lionel.Draghi <Lionel.Draghi@fr.thalesgroup.com>
Date: Fri, 14 May 2004 13:44:52 +0200
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

Lutz Donnerhacke wrote:

> It looks other than C. You can't be productive while learning.

That's absolutely *not* our experience here in Thales Communications. Beginners on my (rather complex) project understand quickly the code, because Ada is crystal clear. They are able to fix bug quickly because the language and the compiler catch most stupid syntax error.

We have several time successfully integrated beginners (first job and no Ada knowledge). It was easy, provided that peer reviews are done and that experienced programmers are available for sharp questions, and that they start with simple coding task.

Those beginners will know 50% of Ada long before knowing 10% of our domain, and that's a much bigger cost.

From: Martin Dowie <martin.dowie@btopenworld.com>
Date: Fri, 14 May 2004 18:11:13
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

A company I'm familiar with (cough!) take non-computer degree qualified people and train them to be Software Engineers in high-intensity courses. The language used - Ada. Why? See the article at

http://www.computerweekly.com/Article106744.htm

[See also the “Ada and Education” section of the News in this AUJ issue. –su]
Try that with C/C++...

With Eiffel/Java you might stand a hint of a chance.

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: 15 May 2004 00:35:53 +0200
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

[...] Kai Glaesner wrote:

> I think today's avionics tend to more and more based on COTS hardware and rely heavily on concurrency, distribution, being physical connected via Ethernet and using kind of realtime-ORB's as an communication infrastructure.

At Barco we design the displays down to the layout of the boards. Most of the ASICs and processors are COTS, but not much more.

> Do you catch the drift? These are all domains Ada is good in.

Yes. We use Ada for almost everything.

> I don’t think the problem are Ada compilers being targeted to the wrong processor, but software managers in avionic business not knowing about Ada capabilities in the above mentioned architectures. This leads to e.g. UPSAT (now Garmin) deploying a PC104-based(certified) Moving Map Display (the MX20), developed in C/C++ running under Windoz NT(!).

Certified to what level? When we do DO-178B level A or B we certainly do not use Windows XP. If it's level E then it is not life-critical, and the only potential problem is customers complaining they've paid millions in development contracts just to get the blue screen of death :)

From: Jeffrey Carter <jrcarter@acm.org>
Date: Sun, 16 May 2004 16:48:10 GMT
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

Jacob Sparre Andersen wrote:

> Why is it that everybody seems to say that it is slower to get programs written in Ada out of the door, than those written in other languages?
> Are there any data indicating that this is the case?
> Or some sensible arguments for it?

I'm aware of 2 real-world studies, in 2 different domains, of real projects, done both in C and Ada, with good metrics, that show that Ada reaches deployment in half the time/cost of C. As a result, I see no evidence to support claims that C-like languages have better time-to-market characteristics than Ada.

These same studies show Ada has a factor of 4 fewer post-deployment errors, and a factor of 10 less time/cost to correct an error.

Note to MDC: All other things are not equal, but that was true of these projects as well, and Ada still came out with a factor of 2 advantage on time to deployment.

From: Jeffrey Carter <jrcarter@acm.org>
Date: Tue, 18 May 2004 01:05:59 GMT
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

Marin David Condic wrote:

> Keep in mind that I have done a study that showed a 50% reduction in development time and a factor of 4 improvement in defects. You may in fact be thinking of that very study here. I've made the statement here before.

One was a study in the domain of jet-engine control, which I think was presented by some guy named Condic. Is there a link to this study? That would be a nice thing to have. The other is the Rational/Verdix study in the domain of Ada compilers, which is available on AdaIC.org. There are other studies also available on the AdaIC site, such as John McCormick's. They don't present their results in a comparable way, but the general message is the same for all of them.

From: Peter Amey <peter.amey@praxis-cs.co.uk>
Date: Tue, 18 May 2004 08:58:03 +0100
Subject: Re: Ada used in General Aviation (GA) applications?
Newsgroups: comp.lang.ada

On the subject of defect rates, I think data from the Lockheed C130K (Hercules II) project is relevant. Independent V&V was carried out by the UK MoD (the lead customer). The V&V covered a variety of

systems produced by a variety of sub-contractors in a variety of languages. All the software inspected had already been cleared to DO178B level A or B.

The results showed that:

1. Significant errors remained despite the prior FAA clearance.

2. Code written in C had, on average, 10 times as many errors as that written in Ada (and 100 times more than that written in SPARK).

3. No statistically significant difference in error rate between level A versus level B systems could be found.

See: "Software Static Code Analysis Lessons Learned" by Andy German, QinetiQ Boscombe Down. DoD CrossTalk Journal, November 2003.

http://www.stsc.hill.af.mil/crosstalk/2003/11/index.html

and

"Correctness by Construction: Better Can Also Be Cheaper" (PDF 312kb) Peter Amey, Praxis Critical Systems Limited. CrossTalk Magazine, March 2002. http://www.praxis-cs.co.uk/pdfs/c_by_c_better_cheaper.pdf

Ada and COBOL

From: I R T <rambam@bigpond.net.au>
Date: Tue, 08 Jun 2004 09:07:16 GMT
Subject: Re: 7E7 Flight Controls Electronics
Newsgroups: comp.lang.ada

"Richard Riehle" <adaworks@earthlink.net> writes:

> we will have a difficult time digging Ada out of the hole it is now in. It has been suggested we rename the language.

Renaming Ada 200x would probably help to spark some interest.

Right now, people go," Ada, Oh we know all about that <snigger>" when the truth is that they know sweet f*ck all about Ada.

All they know is the usual folklore (slow, clunky, old, obsolete, large, Pascal with COBOL bolted on etc)

Renaming the next version, would spark some interest, but would not by itself be enough to keep interest.

From: Leon Winslow <leon.winslow@notes.udayton.edu>
Date: Sun, 06 Jun 2004 17:37:00 -0400
Subject: Re: 7E7 Flight Controls Electronics
Newsgroups: comp.lang.ada

Jeffrey Carter wrote:

> Marin David Condic wrote:
> The bulk of software development is being done in some other language besides Ada.
> Right. More SW is developed in COBOL than any other language.
> "COBOL? Not *that* dead, old failure!"

There is an irony here that most people might not appreciate. COBOL was the first language mandated by the federal government. I don't know if Ada was the second or there were one or more other languages in between these two.

Some interesting questions might be: Why was one more widely used than the other. What did the government do differently with COBOL and Ada and what effect did it have?

From: I R T <rambam@bigpond.net.au>
Subject: Re: 7E7 Flight Controls Electronics
Date: Mon, 07 Jun 2004 11:08:40 GMT
Newsgroups: comp.lang.ada

Association with the military was the kiss of death as far as many developers were concerned.

COBOL also had the benefit of backing by IBM.

From: Richard Riehle <adaworks@earthlink.net>
Date: Tue, 08 Jun 2004 02:22:05 GMT
Subject: Re: 7E7 Flight Controls Electronics
Newsgroups: comp.lang.ada

Actually, the story is a little different. I recall a large USAF project in the late 1960's where IBM was pushing PL/I, and pushing it hard. At that time, there were seven other mainframe companies that wanted to bid on the contract, but only CDC had anything like a useful PL/I compiler.

The "Seven Dwarfs" spearheaded by Burroughs and Honeywell protested the PL/I requirement and persuaded the USAF to specify COBOL instead of PL/I. IBM, which intended to let COBOL die a slow death in favor of its own language, was forced to revive its COBOL effort. In the end, IBM did not win the contract. As I recall, it was won by Burroughs.

IBM wanted COBOL gone. The only reason it kept it alive was to satisfy RFP requirements from the DoD. In time, COBOL became the dominant language for business data processing, even though IBM continued to insist on the superiority of PL/I.

The fact is that PL/I was a superior language. However, it was so dramatically different in look and feel from the languages it was intended to replace (Fortran and COBOL) that neither language user group found it attractive. If IBM had been successful with PL/I, there would probably never have been an Ada. The fundamental elements were already in PL/I, but it also had a lot of inherent flaws that needed to be rectified before it could be selected.

BTW, the first Alsys Ada compiler was, if I recall correctly, written in PL/I. Ada was, and always has been, a far better language than PL/I, but PL/I could have been improved with a little effort and cooperation from IBM. However, PL/I, at that time, had already earned a widespread bad reputation, much as Ada has today. Overcoming a bad reputation for a language is almost impossible, even when the language has improved as much as contemporary Ada.

I was in a meeting last week where someone commented, "The Chair of our computer science department believes the DoD has banned Ada." With this kind of misinformation as widespead as it is, we will have a difficult time digging Ada out of the hole it is now in. It has been suggested we rename the language. I think it is better to follow Marin's advice and simply build the best systems we can using it. Also, we need to counter the idiotic claims made in ACM's Queue magazine. ["Security: The root of the problem" by Marcus J. Ranum, ACM Queue vol. 2, no. 4 (Jun 2004), available on-line at http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=160 --su]

From: Richard Riehle <adaworks@earthlink.net>
Date: Fri, 11 Jun 2004 18:49:41 GMT
Subject: Re: 7E7 Flight Controls Electronics (COBOL Popularity)
Newsgroups: comp.lang.ada

Warren W. Gay wrote:

> One of the areas that COBOL was very successful in (and still), is providing the necessary facilities to perform business functions. While it may seem trivial, the need to format numeric values (particularly monetary values) in a picture format is so prevalent, that it becomes a major pain to use other languages that don't conveniently provide this.

You wrote this in reply to my note about the role of the DoD in the survival of COBOL. The events of the time, mentioned in my earlier post, were such that IBM had a virtual monopoly in the computer business, much the way Microsoft has today. At that time, the Federal Government was less inclined to accomodate that monopoly than is the current government. Therefore, a lot of effort was made to ensure that all qualified bidders were able to compete for contracts.

For the USAF contract (as with many other contracts at this period of computing history) was originally specified for PL/I. IBM managed to get that requirement into a lot of Requests for Proposal. Protests from several vendors, as well as from the community at large, resulted in that requirement being replaced by COBOL.

It was IBM's intention to replace both Fortran and COBOL with PL/I. If IBM had been successful, the history of computer programming languages would have been much different. PL/I was, in many respects, an improvement over COBOL

and Fortran. However, IBM failed to manage its acceptance by the computing community -- much the way the DoD mismanaged the Ada initiative almost from the start.

> Of course, I am sure there were many other factors that played into the popular use of COBOL besides this. I've forgotten any COBOL I once knew, but I seem to remember that having builtin support of Indexed files etc. to be a great asset to business.

The survival of COBOL is a complex story. Much of that story is political. Some of it is technological. Most of it is due to Newton's First Law of Motion.

> PL/I had a number of whizbang features (for the time), but they didn't exactly pander to the real business needs (I don't recall if the full PL/I language supported the picture formatting or not). Certainly one of PL/I's downfalls was the shear size of the language, for the time.

PL/I did not look like COBOL to COBOL programmers and it did not look like Fortran to Fortran programmers. We all have had to experience of a programmer being resistant to some new language (e.g., Ada), not because it is a bad language design, but because it does not look like the language they are used to. Anyone who has tried to persuade a C programmer to consider Prolog, an Ada programmer to consider C++, a C++ programmer to consider Ada, or a Forth programmer to consider C, understands how this works. PL/I, though it had some small flaws in its original design, had all the elements needed to evolve into a better language than those in widespread use at the time.

This is an Ada forum, so we might take notice of the lessons of PL/I when discussing Ada. In the case of PL/I, and large organization, IBM, tried to bully its customers into using a new and strange language to replace what they were already using. In the case of Ada, a large government agency tried to dictate the use of an equally strange and unfamiliar language. The people who actually develop software resisted, sometimes with malicious compliance, other times with outright defiance, and both the PL/I mandate and the Ada mandate failed. The failure had little to do with the relative virtues of the respective languages.

It had much to do with the fact that human beings dislike being ordered to change their ways.

Now that Ada must be a choice rather than fiat, we have the opportunity to persuade people to use it rather than mandate its use. The only way we can do that is through example. Those who believe it is a superior approach to software development need to prove it by building better software with it. Then they can announce their success with Ada. There is no other avenue for Ada's long-term success. No amount of preaching, complaining about someone else's stupidity, or managerial incompetence will garner a single iota of success. Only success will lead to success. Prove Ada by its fruits, not through declamation and oratory.

How to Persuade a Manager

From: Toshitaka Kumano <kuma@mss.co.jp>
Date: Fri, 9 Jul 2004 15:31:58 +0900
Organization: Mitsubishi Space Software
Subject: Advocacy Needed
To: team-ada@listserv.acm.org

We developed an experimental naval system in the early 1990', which was very large scale real-time system, programmed almost with Ada83, and with a little of C89 / assembler code.

Now we plan to put the system into actual production in coming years, but the manager and some system engineers are very dubious that Ada will survive, say, in a decade from now, and they consider that maintenance problem shall arise, sooner or later, by shortage of various tool chain.

They plan to convert manually the entire sources from Ada83 to C++, that is very ridiculous, from my viewpoint as an evangelist of Ada for its technical superiority, and a believer of the survival of the language.

However, such a report in U.S. like

http://www.sei.cmu.edu/pub/documents/03.reports/pdf/03tn021.pdf

is enough to persuade managers in Japan that "Ada is Dead or Dying".

To persuade manager with technical superiority of Ada is of no use here, because they understand that to some extent, and they simply concern about shortage or soaring price of tools in future.

I need some powerful advocacy among U.S. defense developers that "Ada will Survive", even if it (she?) may be not of mainstream of future information systems.

Any URL to *recently* published report or articles are welcome, but forecast articles from Navy or DoD officials, or articles for some long life-cycle defense system with Ada, would be most powerful for persuasion.

From: Paul Pukite <puk@umn.edu>
Date: Sat, 10 Jul 2004 14:22:47 -0400
Subject: Re: Advocacy Needed
To: team-ada@listserv.acm.org

Forward this email I am sending to your managers. I am a person referenced in that article, based on a web page I have been maintaining over the past 10 years.

If I must tell you one thing: I believe the author Jim Smith lies and distorts the facts. He puts this quote in his paper "In a related observation, Paul Pukite notes a dramatic decrease in the number of Ada articles published by various trade magazines (vice academic journals) from a peak of 27 in 1995 to 1 in 2002 [Pukite 02]."

I said no such thing. I did not speak to the Jim Smith fellow and I have not written this anywhere on the web or in any publication. And bottom-line, I do not believe the statement is even true. You can take a look at my "Ada in the Trade Press" page (at http://umn.edu/~puk) and see that the number of papers written is somewhat constant over the years. But then again even given that statement, I don't pretend to capture all the papers written that contain Ada references. I used to look at more trade journals in the past, but with the advent of web publishing, printed matter is becoming less and less relevant. To top it off, the main thing to remember is that Ada is becoming a commodity product. Commodity products are things like zippers; you just use them and rarely have to worry, thus fewer references (e.g. How often do you see referenced "Zipper World Today"?).

Also, consider this: I have received e-mail from Jim Smith in the past. To whit, this is what I dug off my archives --

---

Paul,

I recently received an e-mail from an individual at Praxis Critical Systems with a list of some of their article which have been published recently...

James D. Smith II (jds@sei.cmu.edu) <http://www.sei.cmu.edu/staff/jds/>

Senior Member of the Technical Staff

Software Engineering Institute

So basically, Smith has an agenda and achieves it via dissembling data. He sent me a friendly email for a new paper to place on my trade-press page, and then turns around and commits the scientific equivalent of slander. I have come to expect a kind of sloppiness in reporting from the technical and mass media, who are often on a strict deadline. But, to have a "scientific" paper, ostensibly from a reputable organization such as CMU-SEI, prevaricate like this is particularly loathsome.

I am seriously considering sending a letter to the government funding agency responsible for sponsoring that CMU-SEI paper outlining my concerns of serious academic misconduct.

Paul Pukite, PhD, Electrical Engineering

From: Peter Amey
<peter.amey@praxis-cs.co.uk>
Date: Fri, 9 Jul 2004 09:00:41 +0100
Subject: Re: Advocacy Needed
To: team-ada@listserv.acm.org

Some arguments that I have used:

1. How does your manager know that C++ will survive? How many truly independent C++ compiler vendors are there?How compatible are their dialects (should your chosen vendor disappear)? At least Ada is truly portable so you only need any one vendor to survive (not a particular one).

2. Some C++ compiler vendors are also Ada vendors and their compilers share large numbers of components so they get maintained together and target new processors together.

3. If all else fails, GNAT is open source and can be kept going indefinitely.

4. What does providing 25 years support mean in practice?

a. In Ada: Language transition: Ada 83 >> Ada 95 >> Ada 0Y; Almost completely backward compatible; Language type abstractions protect from processor size changes

b. In C: Language transition: K&R C >> C90 >> C99, Significant backward incompatibilities; Processor word size dependencies easily introduced; C99 is very large, has major insecurities and no compiler support

c. In today's "Hot Language": Language transition C >> C++ >> Java >> C# >> ???; Near complete rewrite at each stage!

5. It may not happen anyway: if your manager is prepared to pay the price of porting now, when there is no proven risk of future maintainability problems, why is he not prepared to defer that expenditure and make the port only if his worst fears are realised and maintainability does become a problem?

6. Ada is healthier than some of the special pleading reports might suggest. Ridiculous arguments are used against Ada by those with vested interests. We have have posted here "number of open source projects using Ada" being used as a (spurious) metric and elsewhere I have seen ostensibly sane people claiming "we can't use Ada because no university in Texas teaches it". Where reliability is favoured over fashion, Ada continues to be used. The 7E7 Dreamliner will be a substantially Ada aircraft (especially where it matters). Ask you manager whether he regards reliability as important.

7. If all else really fails, Tucker will convert to C for you using Adamagic and you could maintain the C.

I think the main thrusts should be: (1) He is spending money now against an unknown risk and it won't cost any more to do the port in the future if the risk materializes. (2) Ada is healthier than a casual glance suggests.

Hope this helps

From: Wojtek Narczynski
<wojtek@power.com.pl>
Date: Fri, 9 Jul 2004 12:27:54 +0200
Subject: Re: Advocacy Needed
To: team-ada@listserv.acm.org

Isn't it the case that for the money that will go in porting (and testing :-O) you could actually ensure that i.e. GNAT, or your current tool chain of choice, will survive in good shape?

But then, don't worry: there seems to be some Ada research interest... in China, there were some papers, it is not unreasonable to think that they are tip of an iceberg. So if not in US Defense systems, Ada will survive in Chinese ones. So there is really no need to worry about. What were we discussing? A web dating service? Oh no! - a critical defense system. So the #1 question is: does the commander realize that C++ is not nearly as good for this system as Ada?

From: Alan and Carmel Brain
<aebrain@webone.com.au>
Date: Fri, 9 Jul 2004 20:51:06 +1000
Subject: Re: Advocacy Needed
To: team-ada@listserv.acm.org

It depends on the problem domain.

If your problem is to get the thing done as quickly and cheaply as possible, with as much unreliability as you can get away with, and quickness-to-market and most-features-possible (whether useful or not) are critical for success, then you use one technique.

If you want something which people can bet their lives on, then you use another.

Ada is being marginalised into a niche. Only projects such as aircraft and spacecraft avionics use it more than they use other languages.

The trouble is, that all the arguments for using C, C++ etc hold even more for Visual Basic, Visual Basic for Excel etc. There, a new compiler comes out every few months, there are literally millions of cheap programmers available, and vast numbers of tools come out every year to try to get around the languages' basic limitations. This is even more true for assembler: every time a new CPU comes out, a new assembler is made for it, and soon thereafter, tools for assembler code-generation in vast profusion, often C or C++ compilers.

But this is preaching to the converted, and of limited use to the original poster.

Hopefully what I have to say below may help. [...]

The first non-Japanese payload to be carried on board a NASDA rocket was the Australian 'Fedsat' satellite.

The satellite bus was programmed entirely in Ada-95.

The CPU was the ESA-designed ERC-32.

The RTOS was RTEMS, originally designed in Ada, but later ported to C. This is freeware, it cost us nothing.

The application was programmed using the GNAT 1.13p compiler. Free, it cost us nothing.

The tool we used to simulate the ERC-32 was written in Ada, and we used the freeware version, which cost us nothing.

The FedSat satellite had experiments on board including a Magnetometer, a GPS system, a communications payload for mail-forwarding to ocean buoys, a star camera, and a FPGA (Field Programmable Gate Array) experiment. In addition, the Attitude Control System was space-qualified by FedSat, Architecturally, it too was basically another experimental payload

All of these payloads, and the satellite system itself, are seperately re-programmable via telecommand from the ground. Since launch, several have been re-programmed, and FedSat has demonstrated for the first detection and self-repair of radiation damage using FPGAs.

Fedsat is almost certainly the most complex satellite of its size ever launched. Some CPUs used IEEE-format floating point representation, some used 1750 format. The Mass memory used a different endianism from the rest of the satellite, and due to financial constraints, wasn't fully populated: thus ABCDEF in addresses 0-5 would be stored as BAxxDCxxFExx in addresses 0-11, with bank-switching between pages. Coping with this was difficult in Ada-95, but doing so in C or C++, within the very strict and tight time performance constraints, and also preventing starvation or lockup with multiple concurrent processes attempting to use this common resource... would have been impossible given the budget of time and money, especially for a high-rad environment with soft-failures certain.

The total software budget for the satellite (though not the payloads) was on the order of $300,000 USD, and the software team was an average size of 2, with a peak of 3 and a trough of 1. During the final 6 months before launch, the sole programmer involved was a new graduate (who had never used Ada outside one semester at University before starting the project 12 months previously).

If you wish to get more details about the project, feel free to contact me. I headed the Spaceflight Software Development Team, and wrote about 1/3 of the code on board the bus.

Given these metrics - zero cost compiler, zero cost emulation tools, and being able to train a beginner Ada programmer to

become an outstanding one within 12 months, it is very difficult to see any objection to the use of Ada from a maintainability viewpoint. Especially since the last update to the satellite's software was performed by another Ada beginner, one who wasn't on the original team (good documentation really helps with any language, but with Ada it makes maintenance by beginners normal, cheap and reliable)

It may not be true when writing the next video game, nor the next 'killer app' financial spreadsheet, and certainly not when doing GUI prototyping, but when you have to have reliability and reuseability,

Better

Cheaper

Faster

With Ada-95 it's 'Pick any three'. The proof is in orbit, put up there by a Japanese rocket.

From: Martin Dowie
<martin.dowie@btopenworld.com>
Date: Fri, 9 Jul 2004 13:36:48 +0100
Subject: Re: Advocacy Needed
To: team-ada@listserv.acm.org

The first two systems listed in "Crosstalk" (AFATDS and DMLSS) are both Ada projects with a long term future. They were 2 of the "U.S. Government's Top 5 Quality Software Projects".

http://www.stsc.hill.af.mil/crosstalk/2004/07/index.html

Hope this helps

From: Stephen Leake
<Stephe.Leake@nasa.gov>
Date: Fri, 9 Jul 2004 09:34:34 -0400
Subject: Re: Advocacy Needed
To: team-ada@listserv.acm.org

I suggest you contact the Ada vendors. ACT in particular can make a very good case that they will survive. Then ask the C++ vendors the same question.

Ask this question: "which tool vendors will sign a ten year support contract with us". Ada companies will; I'm not sure about C++ companies.

From: Alexander E. Kopilovich
<aek@VB1162.SPB.EDU>
Date: Sat, 10 Jul 2004 05:09:36 +0400
Subject: Re: Advocacy Needed
To: team-ada@listserv.acm.org

It is hard to believe that just vague and uncertain concerns about rather far future can be the real reason for undertaking quite a big and costly work. Perhaps there are other reasons, which are pushing for that move. For example, a perspective of an active development phase (surely, that "manual conversion" will weight at least as a surrogate development) with all its attributes - more people, more alive status, room for changes, etc. If these my suspicions are at least partially true then the desire for the move from Ada should be counteracted using appropriate arguments - those that will stand against exactly that particular direction of the move, but at the same time permit fulfillment of true aims. So, I'd like to add my humble proposition to the Peter Amey's analysis and Stephen Leake's advice.

Tell them (the manager and those dubious system engineers) that their concerns certainly have some ground, but at the same time are heavily overstated. That indeed there is a need for major overhaul of the system because the current trends should be properly (and in good time) reflected. In particular, precautions should be made against the concerns related to possible problems with Ada tools in some future. One cannot guarantee that transition to C++ or other object-oriented language of choice never becomes a necessity. Therefore some steps should be made for making such a transition less difficult and painful. As the current system is programmed in Ada 83, a right plan would be to instill object-oriented character into the system by moving to Ada 95 (perhaps with corresponding renovating changes in system design), and at the same time getting rid of too complicated elements of Ada language by imposing a set of limitations - for example, choosing a subset from the SPARK limitations.

This way the substantial advantages provided by Ada language will be retained, and at the same time important preparations for a possible future transition to another (object-oriented) language will be made.

What can be better than precisely balanced approach? -:)

From: Stephen Leake
<Stephe.Leake@nasa.gov>
Date: Mon, 12 Jul 2004 08:52:46 -0400
Subject: Re: Advocacy Needed
To: team-ada@listserv.acm.org

The "cost" for Ada consists of two parts: installation and support. Installation of the Gnu Ada compiler on most systems consists of getting the open source distribution and following instructions to compile it. Installation of other Ada compilers may involve actual money.

The support for the Gnu Ada compiler does cost money, and is well worth it.

How good is the support for the C++ compiler? I don't know; I've never paid for it.

On a reasonably sized project, the cost of ACT support for Gnu Ada is small. Hmm. Perhaps that just serves to define my notion of "reasonable" :).

[...]

From: PStachour@acm.org
Date: Mon, 12 Jul 2004 22:20:09 -0500
Subject: Re: Ada Advocacy & Compiler Support /C, C++,Ada -- My experience
To: team-ada@listserv.acm.org

I can speak for the cost and benefit of "C" and "C++" compilers vs Ada compilers.

I have worked on projects that use Ada, and ones that use C & C++.

On those projects, we used C/C++ compilers from Sun and from Microsoft. We found bugs in the compilers and run-times in both cases.

In the case of Sun:

Typically it took a month to get our problem "bundled up" to a form where it was reproducible, and send to Sun support. The errors we reported were acknowledged. It usually took about 3 months to get a dot-release or 'patch' for the compiler or runtime. One I particularly remember had to do with the dispatching of sun's dynamic linker when a thread requested a dynamic link.

It took us almost 8 months to get a fix. But we got a fix eventually. Then we had to convince our customer to put that particular version of that particular fix on their systems. [Both earlier and later ones did not work. Also, the one that worked for our C++ problem broke lots of other C++ programs doing threading]

In the case of Microsoft:

Of all the C and C++ bugs I have reported to Microsoft since 1989 (about a dozen), I have *never* gotten a fix that worked. In most cases, they never sent anything [I was not a paying-for-support customer, and got ignored.] In the cases where I was working for a company that was paying for support, we usually got a "fix" in several months. However, the fix we got never fixed the problem.

An example problem was an error in concurrency in the C++ library that meant that multiple threads all making calls on DB interface routines to SQL-Server did, when heavy load was placed on the system, wind up passing the arguments given in one thread to a DB routine acting for a different thread.

I have not had an concurrency problem with any Ada compiler from Any manufacturer since the ones in the 1983-85 time-frame, when Ada and Ada compilers were new.

By-the-way:

Ada's solid types, and items such as discriminated records (since 1983) give me the ability to control the data in my programs to a degree not even envisionable with C/C++/C#.

Most of my data problems are gone by the time I get a clean compile in Ada. In C and C++, I'm still finding data correctness issues in the programs after they've been in the custoemer's shop 3+ years!

Just one person's experience with concurrency and programming languages and the compilers and run-time support. [...]

Does Ada still have a future in industry?

From: kaleunt@dhmail.net
Date: Sun, 13 Jun 2004 20:37:54 +0200
Subject: Ada a t-il encore un avenir industriel?
Newsgroups: fr.comp.lang.ada

[Complete thread translated from French.]

We have to select the programming language for an industrial IT project carried out in my company.

For now, in the same context on a similar project, I have used Ada 83 and I am convinced that Ada is fully adequate for our new project, but I bump against the following arguments from the younger "experts" in my company:

- Ada is less and less used, so that its future availability is at risk. (It is indeed true that the supplier of the compiler that we would purchase should have to still be in business in 10 year and should have to have maintained their product.)

- A further consequence of this reduced market share is the risk of increased difficulty in finding qualified programmers in the future. Moreover, Ada is very seldom taught in school, where they rather use C++ and Java. (Personally, I believe that no programmer should find it difficult to learn Ada: I am an example of that.)

Under all the criteria that we have set (reliability, real-time, program maintainability, readability, interfacing with other languages, etc.) Ada has come out on top and I can prove it, but I don't know what to think about the "perennity" criterion, which is a crucial economic aspect of that project.

What can you tell on this subject? I have worked in my corner (Ada) the last few years and, from a technical stand point, I am fully persuaded of the power of that language, but I have never bothered to understand its market share. Is Ada used in the new projects of large enterprises? Do you have examples of use?

In spite of the subject of this posting, I am not playing naïve, I really need your advice.

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: Sun, 13 Jun 2004 21:02:01 +0200
Subject: Re: Ada a t-il encore un avenir industriel?
Newsgroups: fr.comp.lang.ada

The most part of enterprises that use Ada share the same concern regarding its perennity. They entertain long-term relation with their vendors, which benefit from such situation, in that they can sign long-term contracts with their customers. For this reason, the market of Ada utilities is stable and perennial.

The ultimate assurance obviously comes from the libre software. As the source code of GNAT is free from any restrictions, no GNAT user may ever find him/herself blocked. In the worst case, even if Ada Core Technologies should disappear, what I obviously do not wish, your company could always undertake to maintain GNAT, whether internally or by outsourcing, or else by resorting to the FSF.

The notion that Ada should be on a declining slot is an unfortunate effect of the "Coué method" [autosuggestion, -- su]. The more we let that notion develop, the more it becomes true. In my mind, the key problem arises from the lack of publicity around the language. The suppliers of Ada products definitely are the most responsible for this deficiency, but the user enterprises should also have a vested interest in some publicity that may raise awareness. There definitely is no shortage of arguments, success stories, or prestigious user enterprises.

From: Lionel Draghi <Lionel.Draghi@fr.thalesgroup.com>
Date: Mon, 14 Jun 2004 11:09:50 +0200
Subject: Re: Ada a t-il encore un avenir industriel?
Newsgroups: fr.comp.lang.ada

Our experience in my project is that the learning curve for Ada is one of the best there may be.

Compounding this with the security offered by the language itself allows one to raise newcomers to sufficient proficiency to control existing code in a time that is negligible in contrast with what it takes to master the application domain.

All it takes is to respect common sense, like, for instance, to tweak with care the share of novices and that of experts in the team, to suit the needs of the development phase, and then to integrate on the "experience" variable over the process (for example, as a factor resulting from the performance review of each "expert / novice" pair).

If we leverage on the qualities of the language, e.g. readability, it is obvious that the learning curve shows excellent. In contrast, it still is difficult to persuade of this argument, those that (by way of example) have spent one year to attain a decent understanding of C++ and its typical idioms. They cannot believe that this can be that simple with Ada. They believe that you underestimate the training costs; that's classic.

I do not reproach them; I myself have for a long time underestimate the learning curve of Ada. When things turned out well, I used to think that they were due to an exceptionally clever fellow, but when I have noticed that this was always the case …

In fact, to be very direct, if the learning curve of Ada should pose problems to a lad, then it wouldn't be wise to keep him/her on the project, except for other, more menial tasks.

From: Lionel Draghi <Lionel.Draghi@fr.thalesgroup.com>
Date: Mon, 14 Jun 2004 11:17:29 +0200
Subject: Re: Ada a t-il encore un avenir industriel?
Newsgroups: fr.comp.lang.ada

I forgot to mention that the knowledge of Ada or the lack of it is a marginal cost at the time of recruitment, but I wish to point out that it also is not difficult to find young fellows with Ada training.

The community does not communicate a lot but it is alive and kicking.

How to Write Paranoid Code

From: Ludovic Brenta <ludovic.brenta@insalien.org>
Date: 29 Jun 2004 08:58:42 GMT
Subject: How to write paranoid code
Newsgroups: comp.lang.ada

An old article I stumbled upon:

"Beyond Ada: The First Paranoid Programming Language"

http://paul.merton.ox.ac.uk/computing/paranoid-programming-language.html

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 *Conference announcements for the international Ada community* at: http://www.cs.kuleuven.ac.be/~dirk/ada-belgium/events/list.html on the Ada-Belgium Web site. These pages contain full announcements, calls for papers, calls for participation, programmes, URLs, etc. and are updated regularly.

2004

October 01 — Seminar "SPARK: A Safer Way to Program", Scottsdale, AZ, USA. Topics include: overview of SPARK language and tools in SPARK toolset.

☺ October 04-07 — 18th **International Symposium on DIStributed Computing** (DISC'2004), Amsterdam, the Netherlands. Topics include: distributed programming languages; distributed applications; specification, semantics, and verification of distributed systems; fault-tolerance of distributed systems; cryptographic and security protocols for distributed systems; etc.

October 05-07 — **Conference on Software Testing** (ICSTEST) - UK 2004, Westminster, London, UK.

October 06-07 — **Ada-Deutschland Tagung** 2004, Stuttgart, Germany. Jointly with the **Automotive - Safety and Security 2004 Workshop**, October 6-7, 2004. Topics include (in German): Methoden und Werkzeuge für zuverlässige Softwaresysteme; Beherrschung der Komplexität in SW-Projekten; UML Profile für zuverlässige Software; Vorgehensmodelle und Lifecycle Management von Systemen; Echtzeitsysteme mit Ada; Sichere Software mit Ada; Ravenscar und weitere Sprachprofile; Erfahrungsberichte über Produktivität, Performance und Kosten in Ada-Projekten; Interoperabilität von Ada und anderen Programmiersprachen; Ada in der Ausbildung; etc.

October 07 — Seminar "SPARK: A Safer Way to Program" -- Eastern US, Monroeville, PA, USA. Topics include: overview of SPARK language and tools in SPARK toolset.

October 10-15 — 7th **International Conference on UML Modeling Languages and Applications** (UML'2004), Lisbon, Portugal.

October 18-20 — 5th **Conference for Quality in Information and Communications Technology** (QUATIC'2004), Porto, Portugal. Theme: "Quality: a bridge to the future in ICT". Topics include: Economics of quality: costs and savings; Critical success factors in quality improvement; Quality and legal issues; Reliability, safety and security issues; System extensibility: accommodating evolving requirements; Integrating new and legacy systems; Improving the quality of legacy systems; Product reengineering for quality improvement; Forecasting quality characteristics; Quality perception by customers; ICT teaching quality; etc.

October 18-22 — ACM/IFIP/USENIX **International Middleware Conference** (Middleware'2004), Toronto, Ontario, Canada. Topics include: Distributed real-time and embedded middleware platforms; Reliable and fault-tolerant middleware platforms; Applications of middleware technologies, including telematics, command and control, avionics, and e-commerce; Novel paradigms, APIs, and languages for distributed systems; Impact of emerging Internet technologies and standards on middleware platforms; etc.

October 18-22 — 18th Brazilian **Symposium on Software Engineering** (SBES'2004), Brasília, Federal District, Brazil. Topics include: Component-Based Software Engineering; Design Patterns and Frameworks; Software Engineering Industrial Applications; Software Engineering Tools and Environments; Software Maintenance and Reverse engineering; Software Quality; Software Reengineering; Software Reuse; Software Verification, Validation and Testing; etc.

☺ October 24-28 — 19th Annual **ACM SIGPLAN Conference on Object-Oriented Programming**, Systems, Languages, and Applications (OOPSLA'2004), Vancouver, British Columbia, Canada. Topics include: object technology and its offshoots.

October 24-28 — 3rd **International Conference on Generative Programming and Component Engineering** (GPCE'2004), Vancouver, Canada. Topics include: Generative techniques for Product lines and architectures, Embedded systems, etc.; Component-based software engineering (Reuse, distributed platforms, distributed systems, evolution, analysis and design patterns, development methods, formal methods); Integration of generative and component-based approaches; Industrial applications; etc.

☺ October 25-29 — 6th **International Symposium on Distributed Objects and Applications** (DOA'2004), Larnaca, Cyprus. Topics include: Enabling Technologies, Middleware, Distributed Objects and Applications, etc.

October 25-29 — **International Conference on Practical Software Quality Techniques & Testing Techniques** (PSQT/PSTT'2004 North), Minneapolis, Minnesota, USA

October 31-11/06 — ACM SIGSOFT 2004 12th **International Symposium on the Foundations of Software Engineering** (FSE-12), Newport Beach, California, USA. Topics include: Component-Based Software Engineering; Empirical Studies of Software Tools and Methods; Generic Programming and Software Reuse; Software Engineering and Security; Software Engineering Tools and Environments; Software Metrics; Software Reliability Engineering; Software Safety; Specification and Verification; etc. Deadline for early registration: October 1, 2004

November 02-05 — 3rd **International Symposium on Formal Methods for Components and Objects** (FMCO'2004), Leiden, the Netherlands.

November 02-05 — 15th IEEE **International Symposium on Software Reliability Engineering** (ISSRE'2004), Saint-Malo, Bretagne, France. Theme: "Achieving Software Dependability through Model-Driven Engineering"

November 04-06 — 2nd Asian **Symposium on Programming Languages and Systems** (APLAS'2004), Taipei, Taiwan. Topics include: semantics and theoretical foundations; type systems and language design; language interpreters and compilers; program analysis, optimization and transformation; models and tools for parallel and distributed systems; language support for security and safety; domain-specific languages and systems; storage management techniques; software development methods and systems; techniques for embedded and mobile code; process algebra and concurrency; etc. Deadline for early registration: October 8, 2004

November 08-12 — 6th **International Conference on Formal Engineering Methods** (ICFEM'2004), Seattle, USA. Topics include: applications in real-time, hybrid, and critical systems, etc.; techniques for verification and validation, theorem proving; links with development methodologies, tool environments, emerging technologies; etc.

November 09-12 — 11th **Working Conference on Reverse Engineering** (WCRE'2004), Delft, the Netherlands. Topics include: Program analysis and slicing; Program transformation and refactoring; Legacy systems; Transitioning to software product lines; Documentation generation; Preprocessing, parsing and fact extraction; Reengineering patterns; Traceability recovery; Reverse engineering economics; UML and roundtrip engineering; Program comprehension; Reverse engineering tool support; etc. Includes the following event:

> November 09 — **International Workshop on Software Evolution Transformations** (SET'2004). Topics include: Parsing and analysis techniques for handling source code; Techniques for round trip synchronization between models and source code; Refactoring source code for improved quality; Experience, lessons learned and suggestions for integrating code transformation approaches into development processes; Transformation techniques for migrating existing systems to new platforms; etc. Submission deadline: October 4, 2004

November 10-12 — European **Software Process Improvement Conference** (EuroSPI'2004), Trondheim, Norway

♦ November 14-18 2004 **ACM SIGAda Annual International Conference** (SIGAda'2004), Atlanta, Georgia, USA. Topics include: safety and high integrity issues, real-time and embedded applications, Ada & software engineering education, Ada in other environments such as XML and .NET, Ada and other languages, metrics, standards, analysis, testing, validation, and quality assurance, etc. Deadline for early registration: October 25, 2004

November 18-19 **CoLogNet / Formal Methods Europe Symposium on Teaching Formal Methods 2004**, Gent, Belgium. Topics include: experiences of teaching Formal Methods, both successful and unsuccessful; educational resources including the use of books, case studies and the internet; the integration, or otherwise, of FMs into the curriculum; the advantages of FM trained graduates in the workplace; changing attitudes towards FMs in students, academic staff and practitioners; etc.

November 19 **XP Day Benelux** 2004, Mechelen, Belgium

November 22-26 John Robinson & Associates - Public Ada Programming Course, Cheltenham, UK. A practical course with two streams covering Ada 83 and Ada 95.

November 30-12/02 17th **International Conference on Software & Systems Engineering and their Applications** (ICSSEA'2004), Paris, France

November 30-12/03 11th **Asia-Pacific Software Engineering Conference** (APSEC'2004), Busan, Korea. Topics include: Software Design Methods, Software Testing, Verification and Validation, Program Analysis, Software Maintenance, Software Development Methodology, Metrics and Measurement, Software Quality Assurance, Object-Oriented Technology and Design Patterns, Components Based Software Engineering, Product Line Engineering, Distributed and Parallel Software Engineering, Embedded & Real-Time Software Systems, Standards and Legal Issues, Software Engineering Education, etc.

☺ December 05-08 25th **IEEE Real-Time Systems Symposium** (RTSS'2004), Lisbon, Portugal. Topics include: QoS support; Real-time systems middleware; Security and survivability; Real-time and dependability; Compiler support; Embedded operating systems; Software engineering; RT programming languages; Scheduling; Formal methods; Case-studies; etc.

December 06-08 6th **Symposium on Operating Systems Design and Implementation** (OSDI'2004), San Francisco, California. Topics include: distributed systems, embedded systems, etc.

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

☺ December 13-15 2nd **International Symposium on Parallel and Distributed Processing and Applications** (ISPA'2004), Hong Kong, China. Topics include: Parallel/distributed system architectures; Parallel/distributed algorithms; Reliability, fault-tolerance, and security; Performance evaluation and measurements; Tools and environments for software development; Distributed systems and applications; High-performance scientific and engineering computing; etc.

December 13-15 **Asia Pacific Conference on Parallel & Distributed Computing Technologies** (ObComAPC'2004), Vellore, India. Topics include: Real-time Parallel and Distributed Systems, Distributed computing, Parallel and Distributed Systems, Compilers for High Performance Parallel and Distributed Applications, etc.

☺ December 15-17 8th **International Conference on Principles of Distributed Systems** (OPODIS'2004), Grenoble, France. Topics: theory, specifications, design and implementation of distributed systems, including: real-time and embedded systems; distributed and multiprocessor algorithms; communication and synchronization protocols; self-stabilization, reliability and fault-tolerance; specification verification of distributed systems; security issues in distributed computing and systems; etc.

2005

January 03-06 — *Software Technology Track* of the 38th **Hawaii International Conference on System Sciences** (HICSS-38), Big Island of Hawaii, USA. Includes mini-tracks on: Strategic Software Engineering; Adaptive and Evolvable Software Systems: Techniques, Tools, and Applications; etc.

February 07-11 — 4th **International Conference on COTS-Based Software Systems** (ICCBSS'2005), Bilbao, Spain.

February 22-23 — IEEE **International Conference on Software - Science, Technology & Engineering** (SwSTE'2005), Herzliyah, Israel. Deadline for submissions: October 1, 2004 (full papers), November 15, 2004 (tutorials, industrial track, business process track), December 15, 2004 (doctoral symposium).

February 23-27 — 36th **ACM Technical Symposium on Computer Science Education** (SIGCSE'2005), St.Louis, Missouri, USA. Topics include: Distributed/Parallel Computing, Formal Methods/Theory, Programming Languages/Paradigms, Software Engineering, etc. Deadline for submissions: November 12, 2004 (birds of a feather, faculty posters)

March 09-11 — 11th **International Conference on Languages and Models with Objects** (LMO'2005), Bern, Switzerland. Topics include: object-oriented programming, components and distributed objects, object engineering, etc. Submissions deadline: October 1, 2004 (abstracts), October 8, 2004 (papers)

March 13-17 — 20th **ACM Symposium on Applied Computing** (SAC'2005), Santa Fe, New Mexico, USA. Includes tracks on: Embedded Systems (topics include: RTOS for embedded systems, Hardware/software support for real-time applications, Compilation strategies for performance enhancement vs. footprint control, Program parallelization for embedded systems, Case studies, etc.); Programming Languages (topics include: Compiling Techniques, Language Design and Implementation, Practical Experiences with Programming Languages, Program Analysis and Verification, etc.); Software Engineering (topics include: Software Reuse and Component-Based Development, Software Reliability and Software Fault Tolerance, Reengineering, Reverse Engineering and Software Maintenance, Real-Time Embedded Systems, etc.), etc.

March 14-18 — 4th **International Conference on Aspect-Oriented Software Development** (AOSD'2005), Chicago, Illinois, USA. Deadline for submissions: October 7, 2004 (workshops, tutorials), October 11, 2004 (demonstrations, practitioner reports)

March 21-23 — 9th **IEEE European Conference on Software Maintenance and Reengineering** (CSMR'2005), Manchester, UK. Theme: "Maintaining for Integration". Topics include: Evolution, maintenance and reengineering; Experience reports (successes and failures); Migration, wrapping and interfacing legacy systems; Reverse engineering of embedded systems; etc. Deadline for submissions: 8 October 2004

March 29-04/01 — **Australian Software Engineering Conference** (ASWEC'2005), Brisbane, Australia. Deadline for submissions: 1 October 2004 (tutorials, workshops, research abstracts), 15 October 2004 (research papers), 4 February 2005 (extended abstracts for industry experience reports)

April 02-10 — **European Joint Conferences on Theory and Practice of Software** (ETAPS'2005), Edinburgh, Scotland, United Kingdom.

- April 03 — 4th **International Workshop on Compiler Optimization Meets Compiler Verification** (COCV'2005). Topics include: optimizing and verifying compilation, and related fields such as translation validation, certifying and credible compilation, but also programming language design and programming language semantics. Deadline for submissions: November 26, 2004

☺ April 04-08 — **International Parallel and Distributed Processing Symposium** (IPDPS'2005), Denver, Colorado, USA. Topics include: Applications of parallel and distributed computing; Parallel and distributed software, including parallel programming languages and compilers, operating systems, middleware, libraries, programming environments and tools; etc. Deadline for submissions: October 8, 2004

April 04	3rd **International Workshop on Parallel and Distributed Systems: Testing and Debugging** (PADTAD'2005). Topics include: optimizing and verifying compilation, and related fields such as translation validation, certifying and credible compilation, but also programming language design and programming language semantics. Deadline for submissions: November 15, 2004
☺ April 05-08	2nd Jahrestagung Fachbereich "Sicherheit - Schutz und Zuverlässigkeit" (2nd Annual **Conference on Safety and Security**), Regensburg, Germany. Event includes: special session "Informationssicherheit im Automobil", workshop "Privacy Respecting Incident Management", etc. Contributions welcome in English or in German. Topics include (in German): Vertrauenswürdige Softwarekomponenten; Zuverlässigkeit und Fehlertoleranz in Hardware- und Softwaresystemen; Formale Techniken, Modellierung, Spezifikation und Verifikation; Betriebssicherheit unter extremen Bedingungen; Standards und Normung; Sicherheitsevaluation und -zertifizierung; Kosten von Sicherheit; etc. Deadline for submissions: October 15, 2004
April 06-08	**Software & Systems Quality Conferences** (SQS'2004), Düsseldorf, Germany. Event includes: ICSTEST International Conference on Software Testing, SQM, a congress focussing on Software Quality Management, and CSVHC Conference on Software Validation for Health Care.
April 10-13	**The Conference for Software Practice Advancement** (SPA'2005), Wyboston, Bedfordshire, England. Topics include: Languages; Distributed, component-based development; Pervasive or embedded systems; Patterns and pattern languages; Lessons learned/experience reports; etc.
April 18-21	Annual **Systems and Software Technology Conference** (SSTC'2005), Salt Lake City, Utah, USA
☺ May (dates still TBD)	**DAta Systems In Aerospace** (DASIA'2005), Edinburgh, Scotland, UK
☺ May 15-21	27th **International Conference on Software Engineering** (ICSE'2005), St Louis, Missouri, USA. Topics include: Software architectures and design; Software components and reuse; Software security; Software safety and reliability; Reverse engineering and software maintenance; Software economics; Empirical software engineering and metrics; Distribution and parallelism; Software tools and development environments; Programming languages; Object-oriented techniques; Embedded and real-time software; etc. Deadline for submissions: October 4, 2004 (tutorial and workshop proposals), December 6, 2004 (doctoral symposium, research demonstrations)
May 22-25	5th **International Conference on Computational Science** (ICCS'2005), Atlanta, USA. Theme: "Advancing Science through Computation". Deadline for submissions: November 1, 2004 (workshop proposals), December 6, 2004 (full papers)
June 06-10	25th **International Conference on Distributed Computing Systems** (ICDCS'2005), Columbus, Ohio, USA. Sponsored by The IEEE Computer Society Technical Committee on Distributed Processing. Topics include: Fault Tolerance & Dependability, Middleware, Real-time & Embedded Systems, Security, Formal Verification, etc. Deadline for submissions: October 1, 2004 (paper registrations), October 8, 2004 (papers)
June 13-17	17th **Conference on Advanced Information Systems Engineering** (CAiSE'05), Porto, Portugal. Topics include: Model and Software Reusability, Distributed and Open Architectures, Languages for IS, etc. Deadline for submissions: October 30, 2004 (workshops), November 30, 2004 (papers, panels, tutorials), February 26, 2005 (posters)
♦ June 20-24	10th **International Conference on Reliable Software Technologies - Ada-Europe'2005**, York, UK. Sponsored by Ada-Europe, in cooperation with ACM SIGAda (approval pending). Deadline for submissions: November 7, 2004 (papers, tutorials, workshops)
June 27-29	10th Annual **Conference on Innovation and Technology in Computer Science Education** (ITiCSE'2005), Monte de Caparica, Portugal. Deadline for submissions: November 15, 2004 (papers, panels, tutorials, working group topics), February 14, 2005 (tips & techniques summaries, posters, demos), April 18, 2005 (working group membership application)
July 18-22	13th **International Symposium of Formal Methods Europe** (FM'2005), Newcastle upon Tyne, UK. Topics include: introducing formal methods in industrial practice (technical, organizational,

	social, psychological aspects); reports on practical use and case studies (reporting positive or negative experiences); tool support and software engineering; environments for formal methods.
☺ July 25-29	19th **European Conference on Object-Oriented Programming** (ECOOP'2005), Glasgow, UK
August 29-09/02	13th **IEEE International Requirements Engineering Conference** (RE'2005), Paris, France. Deadline for submissions: February 7, 2005 (abstracts), February 14, 2005 (papers), March 11, 2005 (workshops, tutorials, panels), April 28, 2005 (doctoral symposium, posters, research demonstrations)

Update on Ada Standardization

James W Moore

Convener, ISO/IEC JTC 1/SC 22/WG 9

The MITRE Corporation, 7515 Colshire Drive, H505, McLean, VA 22102, USA; Tel:+1.703.883.7396; email: James.W.Moore@ieee.org (The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author.)

Abstract

In a previous article, I described the process for updating the standard Ada language specification. This article is a brief update describing the progress made at the June 2004 meeting of the responsible standards committee, ISO/IEC JTC 1/SC 22/WG 9.

Status of the Amendment

ISO/IEC JTC 1/SC 22/WG 9 is responsible for the standardization of Ada. It performs this important job in two meetings a year, each one about 4 hours in duration. Of course, this brevity is made possible by delegating work to subgroups. The subgroup responsible for the Ada amendment is the Ada Rapporteur Group (ARG), chaired by Pascal Leroy. The ARG meets more frequently than WG9, and its meetings are several days in length.

As the time of writing this article (July 2004), the preparation of the amendment is on schedule provided by the instructions to the ARG [1]. As required by the schedule, the ARG Chair, Pascal Leroy, provided the June 2004 meeting of WG9 with a statement of the scope of the amendment [2]. This scope was approved by WG9 at its meeting.

Approval of the scope defines the boundaries of the amendment but does not determine the technical approach to be taken. In short, the scope defines the problems to be solved but not the solutions. The solutions are agreed by approving "Ada Issues" (commonly called AIs). Ultimately, the amendment project editor, Randy Brukardt, drafts the text of the amendment based on the approved AIs. The amendment is then approved via successive voting by WG9 and its parent organizations, SC22 and JTC1.

Many of the AIs needed to write the amendment have been approved and more were approved at the June meeting. However, several issues remain open. The remainder of this article will discuss a few of the issues—some settled and some open.

Secondary Standards

The instructions given to the ARG discouraged the use of "secondary" standards. Experience indicates that vendors only pay attention to facilities that are defined in the base language standard. For example, the advanced arithmetic facilities provided by ISO/IEC 13813 and 13814 were not widely implemented. Upon examining the issues, WG9 long ago decided to recommend the withdrawal of 13814 but to recommend that the content of 13813 be substantially incorporated into the amendment. The scope of the amendment reflects this decision.

On the other hand, WG9 recognized that Ada users desire "standard" bindings to various so-called Application Program Interfaces (APIs), but that these APIs sometimes come and go more quickly than can be accommodated by the formal standardization process. Two years ago, WG9 made an arrangement with the two leading Ada professional societies, Ada-Europe and ACM SIGAda, offering to recognize APIs developed and registered by those organizations. The ARG was invited to refer some AIs to this less formal process. The offered scope statement, though, shows that the ARG decided against this step. The only existing well-defined proposals fit easily within the amendment and there is no need to refer them elsewhere.

Unreserved Keywords

When one extends the functionality of a language, it is often necessary to add keywords describing the new functions. These new keywords can introduce an incompatibility, though. If the keywords are reserved, then programs can no longer use those words as identifiers. Existing programs that use those identifiers no longer compile and execute as before. It is possible to define a language that does not reserve its keywords—PL/I was an example—but experience shows that it is difficult to write parsers for them that provide good diagnostics. Nearly all modern languages reserve their keywords despite the compatibility problems that arise with language revision.

The language changes envisioned by the ARG anticipate three new keywords—**interface**, **overloading**, and **synchronized**. (Another proposal, still under consideration, might add a fourth.) The ARG has suggested that these keywords should be exceptions to the general rule and not be reserved. There is some disagreement with this decision, based on principles of conceptual simplicity. It can be anticipated that WG9 will have to settle this issue by choosing among competing proposals. The decision will probably be made in the summer of 2004.

Character Sets

Life for language designers was a lot simpler when characters sets were confined to fit into an 8-bit code space.

The result was character sets easily usable in English-speaking countries, but clumsy and inadequate in other cultures. In the past two decades or so, a lot of work has been done with so-called "wide characters"—16 and 32-bit encodings that allow gigantic character sets. Large character sets permit the direct representation of the ideograms of some Asian languages as well as direct representation of western characters with diacritical marks. Nevertheless, there is no end of problems. To give an easy example—in French, diacritical marks are sometimes omitted on capital letters. So, is the first letter of Eté (été), the same character or a different character than the first letter of Espace (espace)? The answer is important if you want to do text searching. Another example—the Ukrainian alphabet has a letter ї; is it the same or different than the letter i with an umlaut?

Given that this problem amounts to cataloguing and characterizing (no pun intended) every character of every language in the world, it is not surprising that there are some differences in approach and some disagreement. Unfortunately, WG9's parent, SC22, is at the center of much of the disagreement. In the worst possible scenario, the Ada amendment could become trapped in an argument in SC22, and the argument's resolution could require substantial change to the amendment and substantial impact to our schedule.

We have adopted an approach to mitigate this risk. As usual, we are carefully building consensus among all parties in the ARG and WG9; the Japanese representative, Kiyoshi Ishihata, has been extremely helpful in this matter. We plan to outline our approach and describe it in a short paper to be sent to SC22. We will ask SC22 to either endorse our approach or tell us what other approach to take. In this manner, we hope to have an acceptable answer before we begin drafting the text of the amendment.

The Way Forward

During the coming months, the ARG will continue to work on the AIs and begin drafting the text of the amendment. The next milestone is spring of 2005 when the ARG is requested to submit the draft text of the amendment. It will be circulated to WG9 for comment and consideration by the national bodies. Approval of text would be done by an email ballot during mid-2005.

At that point, the process becomes more formal and proceeds to voting by SC22 and then JTC1. We hope to process the standard through balloting by the parent bodies at the ISO "speed of light" that was described in my previous article. So it is important that all technical issues are resolved within the ARG and confirmed by WG9 prior to progressing the amendment to the subcommittee level. In the past, WG9 has been very successful in anticipating the concerns of SC22 and JTC1 and communicating our technical judgment to them via the national bodies represented in WG9. It is reasonable to hope that the amendment can be approved without further changes and published early in 2006.

Summary

An amendment to the Ada language standard is being prepared for publication early in 2006. It will address many of the issues of usage that have arisen since the language's revision in 1995 but which could not be addressed by the 2001 corrigendum. The amendment project is currently on schedule.

References

[1] ISO/IEC JTC 1/SC 22/WG 9, *Instructions to the ARG for Preparation of the Amendment to ISO/IEC 8652,* WG9 document N412, 10 October 2002. http://www.open-std.org/jtc1/sc22/wg9/n412.pdf. Republished in Ada User Journal, 25:1, March 2004, page 27.

[2] Pascal Leroy, *Proposal for Defining Scope of Amendment to ISO/IEC 8652:1995,* WG9 document N437, March 2004. http://www.open-std.org/jtc1/sc22/wg9/n437.pdf. Republished in Ada User Journal, 25:1, March 2004, pages 30-31.

Developing a Web server in Ada with AWS

Jean-Pierre Rosen

Adalog,19-21 rue du 8 mai 1945, 94110 Arcueil, FRANCE; Tel:+33 1 41 24 31 40; email: rosen@adalog.fr

Abstract

This paper is a summary of the tutorial that was presented at the Ada-Europe conference 2004. It is an introduction to AWS, the Ada Web Server. It presents its main features and functionalities

Keywords: AWS, World Wide Web, Ada.

1 Introduction

AWS (*Ada Web Server*) is a project developed by Pascal Obry and Dmitriy Anisimkov. It started in January 2000, and has now achieved a high level of maturity.

AWS is a complete Web framework, available as free software under the GMGPL (*GNAT Modified GPL*): this means that it can be freely used, even in proprietary software, without requiring the software to be itself covered by the GPL. It can be downloaded from http://libre.act-europe/aws/. It used to depend on the Gnat compiler, but the latest version has been made compiler independent.

AWS is available for Windows®, GNU/Linux, FreeBSD, and other systems. The different versions for various systems are 100% compatible[1].

2 Internet

Internet is a network that allows communication between computers around the world. A full presentation of Internet would fall out of the scope of this paper, we will just remind the reader of some issues that are relevant to AWS.

The World Wide Web (or the Web, in short) technically refers to applications that communicate in client/server mode using the HTTP protocol [1], [2].

Under HTTP, a client sends a request by providing a URI (*Unified Resource Identifier*), to which the server provides a response as a text message. The format of a URI is as:

```
http://www.site.com:8650/dir/name?P1=V1&P2=V2
```

where "http" identifies the protocol, "www.site.com" is the name of the computer, "8650" (optional) is the port number, and "/dir/name?P1=V1&P2=V2" is a string passed to the server which interprets it as it pleases. For example, "/dir/name" can be interpreted as a file name, or as a request name. In the latter case, the optional "?P1=V1&P2=V2" part is generally interpreted as parameter associations for the request (i.e. parameter "P1" has value "V1" and parameter "P2" has parameter "V2".

In the case of AWS, the whole HTTP protocol is managed by AWS. The site and port part identify your computer (but you can tell AWS which port to use). It is important to note that the rest of the URI is just a string, but AWS provides facilities to parse it with the usual meaning.

3 AWS basics

3.1 Basic behaviour

AWS is basically an HTTP server. As such, its operation can be described as follows:

- It opens the HTTP (or HTTPS) message and parses its content.
- It calls a user-defined procedure that will provide the reply to the URI request.
- It encapsulates this reply and sends it back to the client browser.

In practice, you declare an object of type Server.HTTP and start the server by calling the following procedure:

```
procedure Start (Web_Server : in out HTTP;
                 Callback   : in Response.Callback;
                 Config     : in AWS.Config.Object);
```

The response callback is a user provided function that matches the following definition:

```
type Callback is access
function (Request : Status.Data) return Response.Data;
```

Compared to regular web programming, the callback function is like the scripts that you would write in Perl or any other language, but in a sense the scripting language is now Ada.

Once a server is started, the main program should not exit. To this effect, three "Wait" procedures are provided, as shown below:

```
Server.Wait (Server.Q_Key_Pressed);
-- Wait for the Q key to be pressed

Server.Wait (Server.Forever);
-- Wait forever, the server must be killed

Server.Wait (Server.No_Server);
-- Exit when there is no server running (all of them
-- have been stopped)
```

A server can be stopped by calling the Shutdown procedure. Note that it is possible (and in fact quite common) to call Shutdown from a call-back function while the main program is on wait.

[1] We developped an application partly under Windows, partly under GNU/Linux, and never found a single difference in behaviour between the versions.

The main job for developing a web server is therefore to define the call-back function. A typical function would look like this:

```
function Service (Request : in Status.Data)
  return Response.Data
is
  URI : constant String := Status.URI (Request);
begin
  if URI = "/givemethat" then
    return Response.Build
              (Content_Type => "text/html";
               Message_Body => "<p>Hello there !");
  elsif ...
```

Note that since you may have several requests at the same time, the call-back procedure must be thread-safe.

3.2 Accessing the form's parameters and building the response

In general, you'll want to build a specific response according to the parameter fields in the URI. AWS provides several convenient functions to access the parameters without having to parse the URI, as demonstrated in the following example:

```
function Service (Request : in Status.Data)
  return Response.Data
is
  P_List : constant Parameters.List
           := Status.Parameters (Request);
           -- List of parameters
  N  : constant Natural
       := Parameters.Get (P_List, "count");

  -- Numbers is a list with multiple selections enabled
  V1 : constant String
       := Parameters.Get (P_List, "numbers", 1)
  V2 : constant String
       := Parameters.Get (P_List, "numbers", 2)
begin
  ...
```

Conversely, several functions are provided to build the response, either from a string:

```
function Build
 (Content_Type  : in String;
  Message_Body : in String;
  Status_Code   : in Messages.Status_Code
                  := Messages.S200;
  Cache_Control : in Messages.Cache_Option
                  := Messages.No_Cache)
return Data;
```

or from a file:

```
function File
 (Content_Type : in String;
  Filename     : in String;
  Status_Code  : in Messages.Status_Code
                 := Messages.S200)
return Data;
```

3.3 Object oriented AWS

Alternatively, a tagged type (called a dispatch-handler, or for short, a dispatcher) can be used to define the processing of incoming requests:

```
package AWS.Dispatchers is
  type Handler is abstract new
       Ada.Finalization.Controlled with private;
  procedure Initialize (Dispatcher : in out Handler);
  procedure Adjust   (Dispatcher : in out Handler);
  procedure Finalize  (Dispatcher : in out Handler);
  function Dispatch (Dispatcher : in Handler;
                     Request    : in Status.Data)
  return Response.Data is abstract;
...
```

Similarly, a server can be started by giving a dispatcher object rather than a call-back function:

```
procedure Start
  (Web_Server : in out HTTP;
   Dispatcher  : in      Dispatchers.Handler'Class);
```

The benefit of using this approach is that dispatchers can be extended; for example, many dispatchers provide a function to register a call-back (or another dispatcher) for pages matching a given pattern.

The simplest dispatcher is the Callback dispatcher, which acts as a simple wrapper around a call-back function. More dispatchers will be described later.

3.4 Examples

A very basic Web server, which always replies "Hello world!" to any request, can be written as follows:

```
with AWS.Response;
with AWS.Server;
with AWS.Status;
procedure Hello_World is
  WS  : AWS.Server.HTTP;
  function Service (Request : in AWS.Status.Data)
    return AWS.Response.Data is
  begin
    return AWS.Response.Build
              ("text/html", "<p>Hello world !");
  end Service;
begin
  AWS.Server.Start
      (WS,
       "Hello World",
        Callback => Service'Unrestricted_Access);
  AWS.Server.Wait (AWS.Server.Q_Key_Pressed);
end Hello_World;
```

The use of the Gnat-specific 'Unrestricted_Access attribute here is due to the fact that the call-back function is local to the main subprogram; in real applications, call-back functions are defined in packages, and the regular 'Access attribute will work.

As another example, the call-back function for a static page server, i.e. a server that interprets the URI as a file name and returns the corresponding file, can be written as follows:

```
function Service (Request : in AWS.Status.Data)
  return AWS.Response.Data
is

  URI : constant String := AWS.Status.URI (Request);
  Filename : constant String := URI (2 .. URI'Last);
begin
  if OS_Lib.Is_Regular_File (Filename) then
    return AWS.Response.File
      (Content_Type => AWS.MIME.Content_Type
                                        (Filename),
       Filename     => Filename);
  else
    return AWS.Response.Acknowledge
      (Messages.S404, "<p>Page '" & URI &
                      "' Not found.");
  end if;
end Service;
```

4 The templates parser

When developing a web application, it is important to separate *what* is to be displayed from *how* it is to be displayed. The templates parser is a tool that is very useful to build pages from a *template* that describe the layout of data, while the actual content is dynamically computed. Although the templates parser has been developed in the framework of AWS, it is a stand-alone library that can be useful for other applications.

A template is a text file that contains *commands* and *variables*. A parser is provided, which interprets the template file, executes commands, and replaces variables with their actual values.

4.1 Tags

Variables in a template file are called *tags*, and have the form @_NAME_@. When parsing the file, a translation table associates values to the tags. These values can be scalars, vectors (one dimensional arrays) or matrices (two dimensional arrays). Note that matrices are really vectors of vectors, and rows are not required to have all the same length. Constructors are provided to associate values to tags. For example, the following excerpt builds a translation table with associations of various kinds:

```
procedure Tags is
   use type Vector_Tag;
   use type Matrix_Tag;

   B : constant Boolean      := True;
   V : constant Vector_Tag := +"10" & "30" & "5";
   M : constant Matrix_Tag := +V & V;
   S : constant String       := "a value";

   Translations : constant Translate_Table
     := (1 => Assoc ("TEST", B),
         2 => Assoc ("VECT", V),
         3 => Assoc ("MAT",  M),
         4 => Assoc ("VAL",  S));
```

Typically, given the following template:

```
<HTML>
<P>Hello @_NAME_@</P>
</HTML>
```

the result of calling the following procedure:

```
procedure Simple is
   Translations : Translate_Table
     := (1 => Assoc ("NAME", "Bill"));
begin
   Put_Line (Parse ("simple.tmplt", Translations));
end Simple;
```

will be to print:

```
<HTML>
<P>Hello Bill</P>
</HTML>
```

Tags can have modifiers, either *filters* or *attributes*. The complete syntax of a tag is:

```
@_{FILTER:}Tag['ATTRIBUTE]_@
```

AWS provides a number of filters and attributes; a full list would be too long for this paper, but here are some examples to show the kind of modification they perform:

@_UPPER:VAR_@	Content of VAR is translated to upper-case
@_ADD(3):VAR_@	Add 3 to the content of VAR, taken as an integer.
@_EXIST:VAR_@	Returns "TRUE" or "FALSE", depending on whether a value has been associated to VAR.
@_MATCH("Ada.*"):VAR_@	Returns "TRUE" or "FALSE", depending on whether the string in VAR matches the given pattern
@_VECT'LENGTH_@	Number of elements in vector variable VECT.
@_MAT'MIN_COLUMN_@	Length of the smallest row of matrix MAT.

Note that filters can be combined, as in

```
@_CAPITALIZE:TRIM:VAR_@
```

4.2 Commands

```
@@-- Any text
```

This is a comment, the rest of the line is ignored.

```
@@IF@@ <expression>
 …
@@ELSIF@@ <expression>
 …
@@ELSE@@
 …
@@END_IF@@
```

If the first expression evaluates to "TRUE", the lines that follow the @@IF@@ are included, and all lines until the corresponding @@END_IF@@ are skipped. Otherwise, the lines that follow are skipped, the expression following the next @@ELSIF@@ is evaluated, etc.

Expressions in @@IF@@ statements include the usual comparison operators (=, /=, <, <=, >, >=) and logical operators (and, or, not). Parenthesized sub-expressions are allowed. For example:

```
@@IF@@ @_A_@ = "This" or (@_B_@ = 3 and @_C_@ /= 0)
```

Note that @@IF@@ commands can be nested.

```
@@TABLE@@
<code>
@@END_TABLE@@
```

This command is really an iterator. If the name of a vector (or matrix) tag appears in a table, it is replaced by a value from the vector tag The content is repeated until all vector tags and matrix tags are exhausted; a shorter vector is completed with empty strings.

A @@TABLE@@ command can appear in another @@TABLE@@ command (only one level of nesting is supported). At level 1, the name of a vector provides a value and the name of a matrix tag provides a vector. At level 2, the name of a vector provides a value (new iteration) and the name of a matrix tag provides a value. This is especially handy for filling tables. For example, given the following template:

```
<TABLE border=2>
@@TABLE@@
<TR>
  <TD>@_NAME_@</TD> <TD>@_AGE_@</TD>
</TR>
@@END_TABLE@@
</TABLE>
```

the following procedure:

```
procedure Table is
   use type Vector_Tag;
   Names : constant Vector_Tag
               := +"Jean" & "John" & "Hans";
   Ages  : constant Vector_Tag
               := +"10" & "30" & "5";
   Translations : constant Translate_Table
    := (1 => Assoc ("NAME", Names),
        2 => Assoc ("AGE", Ages));
begin
   Put_Line (Parse ("table.tmplt", Translations));
end Table;
```

will produce:

```
<TABLE border=2>
<TR>
  <TD>Bob</TD> <TD>10</TD>
</TR>
<TR>
  <TD>Bill</TD> <TD>30</TD>
</TR>
<TR>
  <TD>Toto</TD> <TD>5 </TD>
</TR>
</TABLE>
```

which will appear in a web page as follows:

Bob	10
Bill	30
Toto	5

There are also facilities for organizing tables into sections with different presentations, accessing the current line number in a table, etc.

@@INCLUDE@@ filename *[parameters]*

This command allows inserting another file. If parameters are provided, they are accessible in the included file as the special tags @_0_@ (the file name), @_1_@ (the first parameter), etc.

5 AWS advanced

5.1 Sessions

Sessions allow storing user-specific data. Sessions are enabled by a parameter in the Start procedure. A cookie is sent to the client that will include a session number; a number of services are provided that allow retrieving the session number associated to a request, and storing and retrieving arbitrary data using the session number as a key.

5.2 Provided dispatchers

In addition to the basic dispatcher described above, several dispatchers are provided with AWS.

URI dispatcher

This dispatcher associates a call-back function (or another dispatcher) to a specific URI, or to a set of URIs that match a given pattern. This makes it easy to provide virtual pages: a callback function can be associated to a page name, and the dispatcher will return the result of the associated function whenever the page is called.

Page dispatcher

This dispatcher interprets the URI as a file name and returns its content. If no such file exists, It returns the result of parsing the 404.thtml template.

Note that it is quite convenient to have the page dispatcher as the default to a URI dispatcher: this way, if you have a mixture of fixed and dynamic pages in your application, every page not registered explicitely as a dynamic page in the URI dispatcher will be interpreted as a plain file.

Method dispatcher

This dispatcher allows associating various call-back functions according to the method (GET, PUT…) used to request the page.

Virtual host dispatcher

This dispatcher allows to associate various call-back functions according to the host name in the URI. This makes it possible to have several "hosts" (as seen from the URI) corresponding to the same program.

SOAP dispatcher

This dispatcher allows to associate two call-back functions, one for regular HTTP requests, and the other one for SOAP requests. Of course, this makes sense only to support the SOAP interface described later.

5.3 Other responses

A number of response constructors are provided, in addition to the ones that build a response from a string or from a file. These include building a response from an Unbounded_String, a Stream_Element_Array (great to return pictures), redirecting to another location, etc.

5.4 Streams

AWS provides its own notion of streams (not related to the predefined Ada streams). A stream is a (tagged) type derived from Resources.Streams.Stream_Type.

When defining a stream, you must define Read and Write operations. This allows you to store data to be returned in a more convenient (or economical) way than plain strings.

AWS provides two predefined streams: Memory (plain bytes in memory) and Memory.ZLib (compressed).

5.5 File upload

File upload is the process of sending a file from the client to the server. This is accomplished by having an INPUT field in a form with the type SUBMIT.

When AWS receives a request that includes a file, it stores it in a special directory (the upload directory), and adds two parameters to the request: one gives the full pathname to the local file, and the other one gives the full name that the file had on the client side. It is then up to the application to do whatever is necessary with the file.

5.6 Push

Push is updating client data periodically, without client request. It is a technique which is quite costly, because it keeps a socket open permanently for each client. For this reason, it is considered better practice to use the Refresh feature of HTTP (client pull) to keep a page up to date.

AWS however supports push. It offers services to register clients, and to send data either to selected clients or to all clients (message broadcast).

5.8 Status page

There is a special status page which is processed directly by AWS. Its name can be chosen or configured (default is aws_status.html). This page is intended mainly for maintenance, and provides various informations about the inner state of AWS.

The page is built by parsing the template aws_status.thtml (redefinable) with a number of tags containing the information. This allows redefining the way the information is presented.

The same information can be obtained from the program by using the package AWS.Server.Status

5.7 Configuration

Many things can be configured in AWS. The most important parameters can be given in the Start procedure, with appropriate defaults for the other ones. Alternatively, there is another Start procedure that takes a parameter of type AWS.Config.Object, which is a structure describing all adjustable parameters. Operations are provided to set or query every parameter.

There is a default configuration object which is initialized (in this order):

- from a file named aws.ini in the application directory. It allows common initialization for the parameters that are the same for all applications.
- from a file named <progname>.ini, where <progname> is the name of the application. It allows for initialization of parameters that are specific to the application.
- from a file explicitely loaded by the application.

The file has a simple text format, whose description is given in the user manual. Here are some examples of parameters:

Admin_URI	the status page name
Upload_Directory	where to store uploaded files
Max_Connection	number of simultaneous connections
Server_Port	the port to connect to
Certificate	name of certificate file for secure servers

5.8 Authentication

Authentication is the process of identifying a user with a Name/Password pair. The process defined by the HTTP protocol when a page requires authentication is as follows:

- Check if the request includes authentication data (check that the user name is not empty by calling the function Authorization_Name).
- If it is empty, return a 401 response by calling the function Response.Authenticate. This function includes a (string) parameter called the "realm" (a root URL)
- The browser will show a login box and resubmit the request. From then on, all subsequent requests under the "realm" will include authentication data.

There are actually two protocols, both supported by AWS. In the *basic* (insecure) protocol, the password is transmitted as part of the request without encryption. The application can access the password by calling the function Authorization_Password, and check that it is appropriate to the user. This protocol is considered insecure since the password is fully transmitted, but it can be acceptable over an HTTPS connexion (see below), since the whole connexion is then encrypted.

The other protocol is called *digest*. In this protocol, the password is not transmitted, but a MD5 checksum over the password, user name, and some other fields is provided instead. The function Check_Digest can be called with the expected password; it will check that the digest received corresponds to the expected password.

5.9 Logging

AWS provides facilities for logging the server's activity. Automatic logging can be started and stopped at will. In addition it is possible to explicitly write log messages or to create rotating logs.

The format of the log file is the same as the one used by Apache, thus allowing existing tools to be used for analyzing the contents.

5.10 Secure server

Until now, we assumed that the protocol used was HTTP. Alternatively, AWS can be set to use HTTPS, the secured (encrypted) version HTTP. This is done simply by setting the Security parameter of the Start procedure to True.

It is therefore as easy to have a secure server as an insecure one. Which raises the question: why would one use an insecure server? To be honest, HTTPS is slightly slower than HTTP. But the real reason why most web sites using HTTP is that HTTPS is quite difficult to configure… with Apache. Not everyone is using AWS (yet)!

5.11 Mailing

AWS provides an interface allowing it to serve as an SMTP client. This permits sending e-mail messages from an Ada application. Of course, this package is provided as part of AWS, but it can be used from any Ada application, even if it is not a web server.

Here is a short example showing how simple it is to send a mail with AWS:

```
   My_Mailer : SMTP.Receiver
      := SMTP.Client.Initialize ("mailhost.axlog.fr");
   Result : SMTP.Status;
begin
   SMTP.Client.Send
   (Server  => My_Mailer,
    From    => SMTP.E_Mail ("Rosen", "rosen@adalog.fr"),
    To      => SMTP.E_Mail ("Obry",  "pascal@obry.org"),
    Subject   => "Latest AWS news",
    Message => "The tutorial is doing fine!",
    Status    => Result);
```

5.12 And more…

There are still many services provided by AWS that make it easier to develop Web applications. Describing all of them would be too long for this paper, therefore we just give a short list here:

- The directory browser. Allows building pages that represent the content of directories.
- The URL package. Various operations to parse the different parts of a URI, and provide safe encoding of URIs.
- MIME. Provides constants for various MIME types, and services to guess the MIME type of a file from its extension.
- Translator. Allows Base64 encoding and decoding, and ZLib compression/decompression.
- Exceptions. Allows defining call-back handlers for processing exceptions caught by AWS.

5.13 Deployment

A Web server usually includes the program itself, plus a number of additional files: static pages, templates, images, icons… If you want to distribute a Web server, this can make the installation process quite difficult, since it will require various directories, that must in turn be known to the application.

To ease this process, AWS provides awsres, a utility that can compile any file into an Ada package. When a file is requested, AWS will automatically extract it from the compiled package if it is available. It is therefore possible to distribute a full Web server application as a single self-contained executable, without any installation process!

6 Distributed applications with AWS

A distributed application boils down to having several computers that exchange information: some computers (clients) send messages (questions) that can have parameters to other computers (servers), who act according to the received message and provide a response.

This description applies to the WWW: a question is a URI, the response is the page. HTTP can therefore be used to build distributed applications, and AWS provides a number of facilities to make this more convenient.

6.1 Using HTTP for communication

In addition to being a *server*, AWS provides a package for being a *client* (AWS.Client). It includes facilities for sending HTTP requests, and retrieving the corresponding response. It can handle authentication, and the Keep_Alive protocol to maintain a permanent connection.

If you just want two applications to communicate over HTTP, without necessarily interpreting the response as a Web page, AWS provides a mechanism for simple communication. On the client side, the following function is provided (from package AWS.Communication.Client):

```
function Send_Message
  (Server     : in String;
   Port       : in Positive;
   Name       : in String;
   Parameters : in Parameter_Set
                   := Null_Parameter_Set)
return Response.Data;
```

The Name parameter is supposed to be the name of a service, and the parameters are provided as an array of Unbounded_String. On the server side, an instantiation of the generic package AWS.Communication.Server will automatically build a server that will call a user-provided function each time a request is received.

In short, all the user has to provide is the function that deals with the request; all the server management is automatically being taken care of.

6.2 Distributed server

If you plan to have a heavily loaded Web server, it is often useful to distribute the load over several computers. You may also want to have dedicated servers, for example a computer dealing with data base requests, independent from the server that only deals with regular pages.

To this effect, AWS offers a *hotplug* facility. A hotplug is just a regular AWS server, but it is a secondary server. There is a main server that receives requests, and the hotplug server can register itself to the main server by providing a regular expression; every request whose URI matches the regular expression will be redirected to the hotplug. Hotplugs can register and deregister at any time, thus allowing for dynamic load balancing, for example.

6.3 Remote services

Several protocols exist for client/server applications over the internet. AWS provides facilities for dealing with SOAP, LDAP, and JABBER.

SOAP

The Web as we know it allows sending a request and getting the answer as an HTML page. While this is convenient for the human being, the form of the response is inappropriate for a program to extract relevant information. SOAP (*Simple Object Access Protocol*) is a protocol that allows retrieving the same kind of information as regular HTTP, but in a format which can easily be used by a program. It is really an HTTP request, whose question and response messages are in XML format. The format is quite

complicated, but fortunately AWS does all the necessary parsing, making it much easier to use.

Typically, a SOAP request is sent to a URI, with a special HTTP header that identifies it as SOAP, and a SOAP action field, whose interpretation is left to the server. The body of the request (the *payload*) includes a procedure name, together with parameters. The SOAP action can be simply ignored, or used as the name of an "object" to which the procedure applies.

AWS offers various constructors appropriate for building parameters, as well as operations to extract the procedure name and parameter values. It supports both writing a SOAP client and a SOAP server.

For example, given the following Ada function which computes the sum of its parameters:

```
function This_Proc (P1 : in Integer; P2 : in Integer)
return Integer;
```

it can be turned into a Web service by linking a URI to the following call-back function:

```
function SOAP_CB (Request : in AWS.Status.Data)
  return AWS.Response.Data
is
  use SOAP, SOAP.Types, SOAP.Parameters;
  PL : constant Message.Payload.Object
     := Message.XML.Load_Payload
                  (AWS.Status.Payload (Request));
  P  : constant Parameters.List
     := Message.Parameters (PL);
  R  : Message.Response.Object;
  RP : Parameters.List;
begin
  R := Message.Response.From (PL);
  declare
    P1 : constant Integer
       := SOAP.Parameters.Get (P, "p1");
    P2 : constant Integer
       := SOAP.Parameters.Get (P, "p2");
  begin
    RP := +I (P1 + P2, "myres");
  end;
  SOAP.Message.Set_Parameters (R, RP);
  return Message.Response.Build (R);
end SOAP_CB;
```

On the client side, a call to this service would be:

```
  P_Set : Parameters.List := +I (10, "p1") & I (32, "p2");
  P     : Message.Payload.Object;
begin
  P := Message.Payload.Build ("This_Proc", P_Set);
  declare
    R : constant Message.Response.Object'Class
   := SOAP.Client.Call ("http://host:8080/soapdemo", P);
    P : constant Parameters.List
      := SOAP.Message.Parameters (R);
    My_Res : constant Integer
       := SOAP.Parameters.Get (P, "myres");
```

The I() function is a constructor to build SOAP parameters from an Integer value; a parameter list is built, then the payload is built with the procedure name and the parameters, and then the response is obtained by calling the service with the appropriate URI. AWS will handle the protocol, build the message, extract the response from the XML, etc. Finally, the result is extracted from the response.

In practice, if you want to use a SOAP service, you have to know that the service exists (like you have to know the URI of a page to access it), but you also have to discover the procedure name of the service as well as the types of the parameters. To ease this process, you can find on the web WSDL documents. A WSDL document describes, in XML format, a SOAP service. WSDL is a proposed standard, developed jointly by Microsoft and IBM, which is in the process of being adopted by the W3C.

A WSDL document in itself is not easy to read. Fortunately, AWS provides a utility called wsdl2aws which parses a WSDL document and generates automatically client stubs to call the service, and server skeleton if you want to write a server that provides the service. All you have to do is fill some procedures with the code to use (or provide) the service.

LDAP

LDAP is the *Lightweight Directory Access Protocol*. It is a lightweight subset of DAP (X.500 standard), which provides remote access to a hierarchical database.

The database maps a hierarchy of names (called the DN, *distinguished name*) to values. AWS provides a client to query the database; there is currently no facility for deleting or modifying data. A typical usage would look like:

```
Directory : LDAP.Client.Directory := Init (Host);
begin
  Bind (Directory, "", "");
  declare
    Response_Set := Search (Directory,
                            Base_DN,
                            Filter,
                            LDAP_Scope_Subtree,
                           Attributes ("cn", "mail"));
  begin
    -- Iterate through responses
      -- Iterate through attributes
```

More functionalities could be added later. Since AWS is an open project, contributions are welcome.

JABBER

JABBER is a "chat" (immediate messaging) protocol. It allows exchanging messages between users connected to a JABBER server. Currently, AWS offers a partial client support to JABBER. Only functions to check the presence of a user and to send a message to the user are supported.

This interface is sufficient, for example, to send alerts to a connected user when there is a problem in an application (that's what it was developed for). Again, contributions are welcome to provide more functionalities.

7 AWS in practice

What kind of application can be developed with AWS? The purpose of AWS is not to replace Apache (this would require many more features), but rather to provide a new approach for Web applications. Instead of having a huge general Web server requiring many scripting languages for dynamic pages, each Web application is an independent server, where all processing is done in Ada.

Among the benefits of this approach are ease of dealing with concurrent accesses (thanks to Ada tasking feature), ease of distribution, independence between content and presentation (thanks to the template parser), and last but not least, efficiency!

Some examples of applications using this approach include:

- GESEM: the registration system for Adalog's seminars (internal application only, not available on the Internet). See [1] for details.
- WORM: Shared bookmarks system (P. Obry, EdF).
- Gnat tracker: ACT's bug tracking system
- Ada-Russia web site (http://www.ada-ru.org)
- Currency change system. (Dmitriy Anisimkov). Known to process 40 to 50 requests per second!
- … and many more

AWS can also be used for programs that are not fundamentally web servers, but use HTML as a GUI. This is a nice way of having a portable user interface.

AWS is especially convenient for applications where the Web interface is only part of a more general application. Imagine for example a real-time system controlling a plant; it can offer a Web interface to remotely monitor the state of the system, however the heart of the application is really the control system, not the Web interface.

Finally, AWS can be used as a communication channel between parts of a distributed application, without necessarily being a "Web" application in itself.

Conclusion

AWS is more than just a Web server: it is a complete development framework for distributed applications, including but not limited to, Internet applications. It is a mature technology, already in production use. There is an estimated user base of 300 people.

AWS is being actively maintained; version 2.0 was issued very shortly before the Ada-Europe tutorial, and brings a number of improvements and new facilities (like transient pages, a POP client interface, a utility for *generating* WSDL from Ada…), and more are expected to come in the future.

Acknowledgements

Many thanks to Pascal Obry for his help in designing this tutorial and reviewing this paper.

References

[3] ftp://ftp.isi.edu/in-notes/rfc2616.txt

[4] ftp://ftp.isi.edu/in-notes/rfc2617.txt

[5] J-P. Rosen, "Experiences in Developing a Typical Web/Database Application", proceedings of the ACM SIGAda Annual International Conference (SIGAda 2003) , ACM Press, ACM order number 825030, San Diego, USA, December 7-11, 2003

The Charles Container Library

Matthew J Heaney
email: matthewjheaney@earthlink.net

Abstract

Charles is a container and algorithms library for Ada95, modeled closely on the C++ STL. The library provides both sequence and associative containers, and specifies the time and space semantics of each container. Charles is flexible and efficient, and its design has been guided by the philosophy that a library should stay out of the programmer's way.

Introduction

The Charles library provides a suite of containers that fall into two basic categories, which differ with respect to how element order within the container is determined. Containers also vary in their time and space semantics, which allows the developer to choose the container with the properties that best meet the needs of his application. Container types and operations are consistently named (mostly following the conventions used in the predefined Ada library), which makes it easy to change from one container to another.

The elements of a *sequence* container are stored in linear order, at positions directly specified by the user. Each sequence container optimizes the cost of insertions differently. A *vector* is the most space-efficient container, and is specifically optimized for insertions at the back end. A *deque* is slightly less storage efficient, but is optimized for insertions at both the front and back ends. The vector and deque containers both provide random access of elements. A *list* is optimized for insertion at any position, but does not provide random access. (Charles does not have dedicated "stack" or "queue" containers, because you can achieve the same effect by simply appending or deleting items from the end of one of the more general sequence containers.)

The *associative* containers associate a key with each element, and then store elements in order by key. For a *set* container an element is its own key, but a *map* container allows you to index an item by some other arbitrary type (in this sense a map is a generalization of an array). Both hashed and sorted associative containers are provided. A hashed container has an average time complexity that is constant, and a sorted container has a worst case time complexity that is logarithmic. These properties are important for real-time programs, which are designed around worst case execution times, and therefore require predictable behavior.

Containers also have unbounded and bounded forms. The *unbounded* forms have no upper bound on the maximum number of elements that may be stored in the container. Internal nodes of storage (comprising the element and any storage overhead) are allocated from the heap during insertions. Unbounded container types privately derive from type Controlled, and storage nodes are automatically deallocated during controlled finalization. Because the container type is controlled, the unbounded container packages must be instantiated at library level. The *bounded* forms place an upper bound on the maximum number of elements. A bounded container type is implemented as a stack-allocated array, whose size is specified as a discriminant. This requires that the container type be limited, but since the type isn't controlled, a bounded container package may be instantiated at any nesting level.

Containers are important because they provide a means of storing and retrieving elements, but ultimately it is the elements in which we are interested, not the container. An *iterator* is the means by which a container allows its elements to be queried, modified, or otherwise manipulated without exposing the representation of the container. The iterator mechanism in Charles is completely general, and an algorithm written in terms of an iterator works for *any* container with the requisite iterator operations.

Design Philosophy

Library design is difficult, because the designer must arbitrate among many different goals, which are often on conflict. In the case of a container library, these goals include flexibility, generality, efficiency, safety, ease of use, simplicity, and elegance. The guiding philosophy in Charles is that it should be easy to do common things, and possible to do less common things.

The library designer cannot anticipate every user's need, nor can he imagine all the myriad uses to which his library will be put, therefore it's best to provide flexible primitives that can be easily combined. However, flexibility is good but it is often at odds with ease of use. If a library were very flexible but hard to use, then the user's experience would be unpleasant. (Programming is work, but it should be fun work, not drudgery.) One example of ease of use in Charles is that it is only necessary to make a single package instantiation to create a container type.

A library must be at least as efficient as the user could write himself, otherwise he won't use the library. The moral of the story is that performance matters. The sorted containers in Charles are implemented using balanced tree so that insertion time complexity is only logarithmic, and the hashed containers expand automatically to preserve a constant load factor. Many of the containers in Charles also have special-purpose operations for optimizing performance. Vectors allow you to preallocate the internal

array, and hashed maps and sets allow you to specify the hash table size explicitly. The sorted sets and maps have insertion operations that allow you to specify a hint about the position of the element, potentially eliminating traversal of the internal tree.

Abstractions should be safe (of course), but safety is not completely orthogonal to other library design goals. To create an abstraction that is completely unbreakable means you'll have to give something else up, usually either flexibility or efficiency. Charles is designed to maximize flexibility and efficiency, and so it defers to the library user the decision about how best to provide safety. After all, it's possible to make a flexible and efficient primitive more safe (just add layers as necessary), but the opposite is not true. This is just another example of the fact that Charles tries to stay out of your way. My philosophy is to trust the programmer, and to assume that he knows what he's doing.

Static Polymorphism

Charles is first and foremost a *generic* library. There are no (public) tagged types, and containers are not organized into a type hierarchy. The unbounded forms are *privately* tagged, but this is only because derivation from type Controlled is necessary to implement automatic memory management, and to allow non-limited container types to have pass-by-reference semantics. Since class-wide programming with container types is seldom (if ever) necessary, generality in Charles is effected using *static* mechanisms. This is simpler and more efficient.

It helps to make a mental distinction here between instantiating a generic component, and simply using the instantiation. The generic formal regions of components differ, and the instantiator must consider what each component requires of the actual types. For example, a hashed container requires a hash function for the key, and it's up the instantiator to supply a good quality hash function (or otherwise the container will perform badly). If the user needs elements to be sorted (or perhaps because computing a hash value for a key type is impossible, or too inefficient), then the instantiator must supply a generic actual relational operator that obeys *strict weak ordering* of elements.

However, once the component has been instantiated, then the differences between components tend to disappear, since each component has more or less an identical interface. This is the basis of *static polymorphism*, and it's exactly how the I/O packages in the predefined Ada library work. Consider the difference between a sorted set and a hashed set. The generic formal region of a sorted set looks like this:

```
generic
   type Element_Type is private;
   with function "<" (L, R : Element_Type)
     return Boolean is <>;
package Charles.Sets.Sorted.Unbounded is ...;
```

The generic for region of a hashed set looks like this:

```
generic
  type Element_Type is private;
   with function Hash (Item : Element_Type)
     return Hash_Type is <>;
package Charles.Sets.Hashed.Unbounded is ...;
```

Here's an example of how a set would be used:

```
procedure Op
   (Set : in out Element_Sets.Container_Type) is
   I : Element_Sets.Iterator_Type;
begin
   Insert (Set, New_Item => E);
   I := Find (Set, Item => E);
   Delete (Set, Item => E);
end;
```

Which set component was instantiated — the hashed set or the sorted set? You can't really tell from this program text, since the instantiated interfaces are (more or less) the same. Where the components differ is with respect to their performance characteristics. A hashed set has fast lookups, but occasionally there would be spikes in the execution time of insertions, since the hash table must expand to maintain a constant load factor (the ratio of number of element to number of buckets is always less than 1). This requires that all the elements on the old table must be rehashed onto the new table. A sorted set doesn't need to do any of this, because the tree is balanced during every insertion and deletion, but that means insertion is more expensive. On the other hand, hash tables can perform badly in the worst case, especially with a poor hash function, which can render a hash table unusable in a real-time program. (In fact, hackers have exploited this property of hash tables to mount denial-of-service attacks, by flooding a server with UDP packets having carefully-chosen headers that all hash onto the same bucket.)

With respect to container performance characteristics, the library designer must defer to the user on matters of engineering trade-offs. The Charles library therefore provides a suite of containers from which a user may choose. The benefit of this approach is that it allows the *user* to decide which component is best for his application.

Vectors

The canonical container is the vector, which is simply a linear sequence of items. The generic package for vectors has separate generic formal types for the index (which must be discrete) and the element, so a vector is very similar to an array. The vector container supports random access of its elements (meaning that index selection has unit time complexity), and is optimized for insertions at the back end of container. For example:

```
   V : Integer_Vectors.Container_Type;
begin
   Append(V, New_Item => 42);  -- canonical insertion op
```

A vector is implemented using an array, and thus has the best storage efficiency among all the containers. Insertion operations (in the unbounded form) automatically expand the internal array when the vector has reached capacity. However, if you know the ultimate size of the vector prior to insertion, then you can use Resize to perform the allocation in advance, which is more efficient.

For a vector and deque, positions are specified using the discrete index type:

```
Insert (V, Before => I, New_Item => E);
E := Element (V, Index => I);  -- O(1) time complexity
Replace_Element (V, Index => I, By => E);
```

For a vector, insertion at the back end has (amortized) constant time complexity. However, as the insertion position approaches the front end, the time complexity becomes linear because all the elements from the insertion position to the back of the vector have to slide up to make room for the new item. (This is why there is no Prepend operation for vectors, although the effect of such an operation can achieved by using Insert to add the item before the position Index_Type'First).

If you intend to perform several insertions at adjacent index positions, then it's better to specify in advance how many insertions you intend to perform, which is more efficient because the sliding of existing elements only happens once. The operation Insert_N is made especially for this purpose:

```
procedure Copy
  (V : Vector_Subtype;
   A : Array_Subtype;
   I : Index_Subtype) is
 J : Index_Subtype'Base := I;
begin
  --dig the hole first, to make room for the array:
  Insert_N (V, Before => I, Count => A'Length);
  for K in A'Range loop
    --now fill the hole with the array elements:
    Replace_Element (V, Index => J, By => A (K));
    J := Index_Type'Succ (J);
  end loop;
end;
```

We have seen how to use the selector function Element to return the value of the element at an index position. However, returning a copy of the element is not a sufficiently general mechanism, as efficiency issues might prohibit copying the element simply to query its value. For modifying an element, Replace_Element isn't adequate either since that operation only allows you to assign a completely new value to an element. (Consider a container whose elements are another container, to which we want to add an item. Remember that the library is designed specifically to allow container abstractions to be easily composed.) To satisfy the need for direct, in-place manipulation of elements, all the containers in Charles have an additional generic selector function that accepts an access type as a generic formal type, and which returns an access value designating a variable view of the actual element:

```
declare
  function To_Access is new
    Generic_Element (Integer);
  E : Integer renames To_Access (V, Index => I).all; begin
  E := E + 1;  -- modify element in-place
end;
```

In the absence of actual reference types a la C++, renaming the result of a function that returns an access type is the only mechanism available in Ada for manipulating an object in-place. (Another possibility is to use a process procedure that accepts the element as the procedure parameter, similar to how passive iterators are implemented. This is the technique used in the standard container library that will be available in the next revision of the Ada language. See AI-302 for the details.)

A vector guarantees that the internal array is (physically) contiguous, and that it has C convention. Therefore you can use a vector (really, its array) as the argument of an operation that expects a C-style array, something like:

```
   Handles : Handle_Vectors.Container_Type;
begin
  ... -- populate handles vector
  WaitForMultipleObjects  --Win32 API
    (Length (Handles),
     To_Access (Handles, Index => First (Handles)),
     False,
     INFINITE);
```

All of the unbounded containers in the library are nonlimited, and hence support assignment (with normal copy semantics). A vector also has an Assign operation that optimizes assignment. Since a vector is controlled, the assignment operat*or* works by first finalizing the vector, which deallocates the internal array, and then adjusting the vector, which allocates a new array that is a copy of the source. The Assign operat*ion* works by simply copying the source elements onto the existing internal array; no new allocation occurs unless the source is larger the target.

Assignment *copies* from one container to another. However, what is often needed is a way to *move* a container, such that there is no copying. All of the containers have a Swap operation that exchanges the internal data structures of a pair of containers. Swap allows you use a vector like handle for an unbounded array, which can be passed around from vector to vector.

Vector operations only expand the array, and never shrink it. Operations such as Clear and Delete change the internal component that specifies the number of active elements in the array, but not the length of the internal array. To deallocate the array, you can use the "swap trick":

```
procedure Op (V : in out Vector_Subtype) is
    Temp : Vector_Subtype;  -- no internal array
begin
  Swap (V, Temp);
end;
```

Here the internal array of V is moved onto Temp, and V's internal array becomes null (therefore "clearing" V). When Temp is finalized, the array that originally belonged to V is deallocated. To shrink the array, the process is similar:

```
procedure Op (V : in out Vector_Subtype) is
    Temp : Vector_Subtype := V;
begin
  Swap (V, Temp);
end;
```

The Temp vector is a copy of V, with the smallest internal array necessary to store V's elements. Swap then moves the new, smaller array onto V. This is the standard idiom for trimming internal storage of a vector.

Deques

A deque (short for **d**ouble-**e**nded **que**ue) is similar to a vector, with the difference that insertion of an element at the front of the container has only constant time complexity. It is implemented as an unbounded array of pointers to fixed-size blocks, with each page comprising a fixed number of elements. Prepend works by adding new elements to the first page, and decrementing an offset that keeps track of which element on the page is the first active element. When there is no more room on the front page, then a new page is allocated and the offset is reset.

The deque container also allows elements to be inserted (and deleted) at any position, but the time complexity becomes linear as the insertion position approaches the middle of the container. One difference from a vector is that to make room for the new element, the existing elements are moved toward the end of the deque that is closest to the insertion position.

Deques are especially useful when you have a large number of items to insert, but the total number of items cannot be determined prior to insertion. A vector would need to allocate a new, longer array each time it reaches capacity, which requires copying the existing elements onto the new array. But when a deque reaches capacity (really, its last page becomes full), it simply allocates a new page, which does not affect existing items and hence is much more efficient. A deque also doesn't need large contiguous regions of virtual memory as a vector does.

Lists

Lists are optimized for insertions at any position. All insertions and deletions, in the middle or at either end, have constant time complexity. However, random access is not supported, and navigating among list positions has a time complexity proportional to the distance between positions. For referring to positions in the list, we use an iterator:

```
I : constant Iterator_Type :=  Last (List);
E : Element_Type := Element (I);
```

To insert an item at a certain position, we specify the position using an iterator value:

```
Insert (List, Before => Iterator, New_Item => Item);
```

The Find operation returns an iterator value, which we can test against the distinguished iterator value Back (a sentinel that designates the logical element immediately beyond the last position) to determine whether the search was successful:

```
   I : constant Iterator_Type := Find (List, Item);
begin
   if I /= Back (List) then ... --search succeeded
```

When iterating over a list (really, any container), the iterator assumes the value Back when all elements in the list have been exhausted:

```
procedure Op (L : List_Subtype) is
   I : Iterator_Type := First (L);
  J : constant Iterator_Type := Back (L);
begin
   while I /= J loop
      Do_Something (Element (I));
      Increment (I);
   end loop;
end Op;
```

This idiom works even when the list is empty. The successor of Last is Back, and the predecessor of Back is Last (allowing you to back up from the sentinel to the list proper). The sentinel has wrap-around semantics, so that the successor of Back is First, and the predecessor of First is Back. This means you can use a loop similar to the example above to iterate in reverse too.

If you need to move an element within a list, or from one list onto another list, the Splice operation is optimal, because it manipulates the internal node of storage containing the element, rather than the element itself:

```
Target, Source : T_Lists.Container_Type;
I, J : T_Lists.Iterator_Type;
...
Splice (Target, I, Source, J);
```

Here, Splice moves (*not* copies) the item designated by iterator J in list container Source, onto the list Target just prior to the element designated by iterator I. This preserves the identity of element that was moved. Additional overloadings of Splice generalize this behaviour to move an entire range of elements.

The list container also provides operations for reversing the list, sorting the elements, removing duplicates, filtering, and for merging two (sorted) lists. The sort operation is stable (meaning that the relative order among equal items is preserved across a sort).

The library has both singly-linked and doubly-linked list containers. Forward and reverse iteration over a double list is provided, but only forward iteration is possible for single lists. The singly-linked list is more space efficient, however, because the storage overhead per node is smaller. The single list also caches a pointer to the last node, so that appending an item is a constant time operation, thus making the single list attractive for use as a traditional "queue" data structure.

Note that the list containers are "monolithic" data structures, the same as for all the containers in the library. There is no structure sharing, and list assignment is by value, not by reference. The name *list* simply emphasizes the fact that this container is optimized for constant-time insertion and deletion of elements at arbitrary positions. The semantics of this abstraction are *not* the same as for the similarly-named "polylithic" data structure in functional languages as LISP.

Associative Containers: Sets and Maps

In addition to the sequence containers, the Charles library also has sets and maps that associate a key with each element. For a set the element is its own key, but for a map there is a separate key type. For the *sets* and *maps* keys must be unique, while for the *multisets* and *multimaps* keys can be the same.

There is only a subtle difference in how sets and maps are implemented. For a set, the element is its own key, and the set doesn't allocate space for anything other than the element. For a map, the key is separate piece of data, and the map is implemented as key/element pairs. For a sorted container, the elements are stored in a balanced red-black tree, and for a hashed container, the elements are stored in a hash table that expands as necessary to preserve a constant load factor.

In a sorted container, keys are ordered according to a less-than relation passed as a generic formal operation. A hashed container uses a hash function to scatter keys throughout the table. When the capacity is reached, the container automatically expands the hash table in order to maintain the load factor. Hash table lengths always correspond to a prime number, as this produces better scatter.

Insertion of an element into a set is done in the normal way:

```
   Set : Integer_Sets.Container_Type;
begin
   Insert (Set, New_Item => 42);
```

For a map, you must also specify the key associated with that element:

```
   Map : String_Integer_Maps.Container_Type;
begin
   Insert (Map, Key => "Adams", New_Item => 42);
```

The insertion operation is overloaded to return an iterator that designates the element just inserted:

```
procedure Op (Set : Integer_Sets.Container_Type) is
   I : Integer_Sets.Iterator_Type;
   B : Boolean;
begin
   Insert (Set, E, Iterator => I, Success => B);
   if B then ... --newly inserted
end;
```

This example illustrates that insertion into a set or map is conditional, since those containers require that keys be unique. Insert returns an indication of whether the key was already in the container. If the indicator is True, then the key was not in the container, and the element (or key/element pair) was successfully inserted into the container. If the indicator is False, then the key was already present, and the iterator designates the existing key. Note that in a multiset or multimap, insertion is not conditional since those containers allow keys to be the same, and therefore there is no need for a success indicator (because insertion is always succeeds).

In order to store a key in a map, it must have a definite subtype. One issue is that some keys are more naturally represented using indefinite subtypes, e.g. a name having type String. To use the map the client would first have to convert the key to the definite subtype used to instantiate the package, and then call the map operation. However, this is both inefficient and syntactically heavy. Since maps with string keys are nearly ubiquitous, the library provides special map containers that have type String as the key type. The packages also accept a string comparison operation as a generic formal, which would (for example) allow key comparisons to be case-insensitive.

Lookups in a hashed associative container are fast: only O(1) on average (assuming your hash function is good). For a sorted associative container, the time complexity is O(log n). The idiom for finding a key in an associative container is the same as for the sequence containers:

```
procedure Op (Map: Map_Subtype) is
   I : Iterator_Type := Find (Map, Key);
begin
   if I /= Back (Map) then -- key was found
      E := Element (I);
      ...
end Op;
```

For a multiset or multimap, Find returns an iterator designating the first member of the group of equal keys, which are always (logically) adjacent.

To visit the members of a group of equal keys in a hashed multimap (or multiset), there's a generic operation designed specially for this purpose:

```
procedure Op (M : Multimap_Subtype) is
   procedure Process (I : Iterator_Type) is
   begin
      Do_Something (Element (I));
   end;
   procedure Iterate is
      new Generic_Equal_Range (Process);
begin
   Iterate (M, Key => K);  -- visit elements with key K
end;
```

This is an example of a kind of passive iterator. The sorted sets and maps have an operation called Lower_Bound, which returns the smallest key in the container equal or greater than a specified key, and another operation called Upper_Bound, to return the smallest key greater than a specified key. These operations are useful for finding the key in the container that is either equal to a specified key (if the key is in the container) or would be adjacent to it (if the key is not in the container). This behaviour is like a search operation that returns a partial match. If you had a set of test scores, for example, then you could use these operations to, say, iterate over the scores within each tenth percentile, even if there are no elements in the set that have the exact percentile boundary values.

The Lower_Bound and Upper_Bound operations return values that describe a *half-open* range of elements, where the first value is in the range and the second value is immediately beyond the range. We can describe a container this way, as a half-open range of elements that begins with First and ends with Back. (These are the values one would use to apply a generic algorithm over all the elements in a container, since generic algorithms don't know anything about containers.)

Lower_Bound and Upper_Bound are also the operations one would use to walk a group of equivalent keys in a sorted multimap (or multiset):

```
procedure Op (M : Multimap_Subtype) is
   I : Iterator_Type := Lower_Bound (M, Key => K);
   J : Iterator_Type := Upper_Bound (M, Key => K);
begin
```

```
   while I /= J loop
      Process (Element (I));
      Increment (I);
    end loop;
end Op;
```

The sorted containers are implemented using a balanced tree and therefore the time complexity of insertions is O(log n). However, if you have a key and know its nearest neighbour in the container, there is a special version of Insert that accepts this information as a hint about where to begin searching for the key, and if the hint is successful the time complexity is only O(1). This is useful for copying a sequence of sorted items into a sorted set (say), since you can use the result of the current insertion as the hint for the next insertion.

The set also has a nested generic package called Generic_Keys that allows you to get a key-view of the element. This is useful when the element is record, and the order relation for elements is computed from one of the components of the record. The canonical example is an employee set, ordered by social security number:

```
type Employee_Type is record
   SSN : SSN_Type;  -- the key
   ...
end record;
function "<" (L, R : Employee_Type) return Boolean is
begin
   return L.SSN < R.SSN;
end;
package Employee_Sets is
  new Sets.Sorted.Unbounded (Employee_Type, "<");
```

We can now add an employee to the set in the normal way:

```
Employees : Employee_Sets.Container_Type;

procedure Hire is
  E : Employee_Type;
begin
  E.SSN := ...;
  E.Name := ...;
  E.Home_Phone := ...;
  Insert (Employees, New_Item => E);
end;
```

Now suppose we want to look up an employee by his social security number. We can't use Find to do this, since Find accepts type Employee_Type, not SSN_Type. However, SSN_Type is what we used to compute the order relation for Employee_Type, so we can use it as the generic actual type to instantiate Generic_Keys:

```
function "<"
  (L : SSN_Type; R : Employee_Type) return Boolean is
begin
  return L < R.SSN;
end;
package SSN_Keys is
     new Employee_Sets.Generic_Keys  (SSN_Type);
```

Now we can use the key-oriented operations to manipulate the set:

```
procedure Op (SSN : SSN_Type) is
  I : Iterator_Type := SSN_Keys.Find (Employees, SSN);
begin
  if I /= Back (Employees) then ...;
end;
```

This is another example of how a sets and maps are very similar. Really they only differ with respect to where the key is stored.

Iterators

A reusable abstraction should be as flexible, efficient, and safe as possible, but these goals are often in conflict. This tension is particularly acute in the design of iterators, since they must provide access to elements of the container without exposing its representation.

In Charles, iterators have been designed to be both flexible and efficient. However, they confer no greater safety benefits beyond what is provided by built-in access types (which is how they're implemented). Where greater type safety is desired (say, to detect dangling references), then the client can implement that himself using the lower-level primitives provided by the library. To see how container iterators are designed, we can compare them to how access types are used to traverse a simple linked list:

```
declare
  Node : Node_Access := List.Head;
begin
  while Node /= null loop
     Process (Node.Element);
     Node := Node.Next;
  end loop;
end;
```

Substituting iterator primitives for access types, we can rewrite the example like this:

```
declare
   I : Iterator_Type := First (List);
  J : constant Iterator_Type := Back (List);
begin
  while I /= J loop
     Process (Element (I));
     I := Succ (I);
  end loop;
end;
```

This schema can be generalized to work for any container besides a list, and this is in fact how iterators are implemented in Charles. The fact that the iterator is nonlimited and definite means you can implement any other type in terms of the iterator (say, to instantiate a container with an iterator as the element type, or to build a higher-level container abstraction that uses reference counting). This is important because real systems are built from the bottom up, by assembling complex abstractions from simpler primitives.

Iterators follow a *machine model*, which views the container as a sequence of elements that are (logically) contiguous. You use the iterator much like a pointer, to designate "addresses" that can be "dereferenced" to manipulate the element at that address. This generality allows us to write generic algorithms that work for any container, even built-in arrays. For example:

```
generic
  type Iterator_Type is private;
  with function Succ
    (Iterator : Iterator_Type) return Iterator_Type is <>;
  with function Pred
    (Iterator : Iterator_Type) return Iterator_Type is <>;
  with procedure Swap (L, R : Iterator_Type) is <>;
```

```
procedure Charles.Algorithms.Generic_Reverse
  (First, Back : Iterator_Type);
```

The first thing to notice is that the container is *not* imported as a generic formal. This makes the algorithm container-neutral, since the container itself is abstracted-away behind the iterator interface (and therefore any container having that iterator interface will do). Note also that the algorithm is element-neutral too, since the element type isn't imported as a generic formal either. This is more flexible because it allows the algorithm to be agnostic with respect to the binding of iterator to an element.

We have shown how the iterator types in Charles bind to container elements via iterator operations as Element, Replace_Element, etc. But let's see if we can use the algorithm above with an array. The "iterator" in this case is simply the array index, and the iterator operations are local subprograms that bind the iterator to array elements:

```
procedure Reverse_Array (A : in out Array_Type) is
  procedure Swap (I, J : Positive) is
    E : constant ET := A (I);
  begin
    A (I) := A (J);
    A (J) := E;
  end;
  procedure Reverse_Array is
    new Generic_Reverse
      (Iterator_Type => Positive,
       Succ          => Integer'Succ,
       Pred          => Integer'Pred,
       Swap          => Swap);
begin
  Reverse_Array
    (First => A'First,
     Back  => A'First + A'Length);
end Reverse_Array;
```

We could just have easily used this same algorithm to reverse the elements of a vector or a list or even a map. Charles has an entire suite of generic algorithms similar to the example above.

The iterators we have described up to now are technically *active* iterators, meaning that the user controls navigation from one element to the next (by calling Succ or Pred, or Increment or Decrement). There is another class of iterators that are *passive*, meaning that it is the iterator itself which controls element navigation. The user's role is limited to supplying a procedure that accepts an iterator (or an element in the case of vector and deque). Passive iterators take the form of a generic procedure:

```
generic
  with procedure Process
    (Iterator : in Iterator_Type) is <>;
procedure Generic_Iteration
  (Container : in Container_Type);
```

Once you have an iterator, you can do anything else such as query the element, replace its value, etc. For containers that support reverse iteration, there is another passive iterator for visiting elements in reverse order.

I often use a passive iterator as a quick-and-dirty way to print the contents of a container during testing, for example:

```
declare
  procedure Process (I : Iterator_Type) is
  begin
    Put (Element (I), Width => 0);
    Put (' ');
  end;
  procedure Print is
    new Generic_Iteration;  --accept Process default
begin
  Print (My_Container);
  New_Line;
end;
```

Passive iteration is preferred to active iteration, since it is simpler and less error-prone. (A common mistake during active iteration is to forget to increment the iterator, resulting in an endless loop.) Passive iterators are also likely to be more efficient, since if a container knows that it is visiting all elements in sequence, then it can visit the elements in a way that takes advantage of that container's unique representation. This is in fact the case for deques, hashed containers, and sorted containers. A sorted set, for example, implements its passive iterator using a recursive algorithm for tree traversal.

Active iteration is still needed, however, when more than one container must be visited simultaneously, although the approaches can be combined. You can use a passive iterator to visit one container, and use the generic actual process procedure to drive the active iterator visiting the other container. This technique would work if one of the containers were an array (the "passive iterator" in that case would just be a for loop).

Conclusion

The Charles library satisfies the need for a general-purpose library having the right balance between flexibility and ease of use. It allows developers to program at a level of abstraction higher than arrays or simple linked lists, using efficient, composable primitives that have specified behavior. More recently Charles has also served as the basis of the design of the standard container library that will be included in the next revision of the Ada language.

The source code for Charles may be accessed via the CVS repository <http://charles.tigris.org/>. All the sources use the GNAT Modified GPL licensing scheme.

Ada-Europe 2004 Sponsors

ACT Europe
Contact: Zépur Blot
8 Rue de Milan, F-75009 Paris, France
Tel: +33-1-49-70-67-16 Fax: +33-1-49-70-05-52
Email: sales@act-europe.fr URL: www.act-europe.fr

Aonix
Contact: Anne Chapey
66/68, Avenue Pierre Brossolette, 92247 Malakoff, France
Tel: +33-1-41-48-10-10 Fax: +33-1-41-48-10-20
Email : info@aonix.fr URL : www.aonix.fr

Artisan Software Tools Ltd
Contact: Emma Allen
Suite 701, Eagle Tower, Montpellier Drive, Cheltenham, GL50 1TA, UK
Tel: +44-1242-229300 Fax: +44-1242-229301
Email : info.uk@artisansw.com URL : www.artisansw.com

Green Hills Software Ltd
Contact: Christopher Smith
Dolphin House, St Peter Street, Winchester, Hampshire, SO23 8BW, UK
Tel: +44-1962-829820 Fax: +44-1962-890300
Email : chriss@ghs.com URL : www.ghs.com

I-Logix
Contact: Martin Stacey
1 Cornbrash Park, Bumpers Way, Chippenham, Wiltshire, SN14 6RA, UK
Tel: +44-1249-467-600 Fax: +44-1249-467-610
Email : info_euro@ilogix.com URL : www.ilogix.com

LDRA Ltd
Contact: Jim Kelly
24 Newtown Road, Newbury, Berkshire, RG14 7BN, UK
Tel: +44-1635-528-828 Fax: +44-1635-528-657
Email: info@ldra.com URL: www.ldra.com

Praxis Critical Systems Ltd
Contact: Rod Chapman
20 Manvers Street, Bath, BA1 1PX, UK
Tel: +44-1225-823763 Fax: +44-1225-469006
Email : sparkinfo@praxis-cs.co.uk URL : www.sparkada.com

Scientific Toolworks Inc
Contact: Matthew Bergeson
321 N. Mall Drive Suite I-201, St. George, UT 84790, USA
Tel: +1-435-627-2529 Fax: +1-877-512-0765
Email: sales@scitools.com URL: www.scitools.com

TNI Europe Limited
Contact: Pam Flood
Triad House, Mountbatten Court, Worrall Street, Congleton, Cheshire CW12 1DT, UK
Tel: +44-1260-29-14-49 Fax: +44-1260-29-14-49
Email: info@tni-europe.com URL: www.tni-europe.com